NASA Technical Memorandum 4709

Development and Applications of a Rosenbrock Integrator

Alan D. Freed and Ilana S. Iskovitz

JANUARY 1996

National Aeronautics and
Space Administration

NASA Technical Memorandum 4709

Development and Applications of a Rosenbrock Integrator

Alan D. Freed and Ilana S. Iskovitz
Lewis Research Center
Cleveland, Ohio

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Program
1996

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This paper extends "the" classical Runge-Kutta integrator of Kutta (1901) to a Rosenbrock (1963) integrator. This extension is analogous to introducing an additional term into a series solution to improve its accuracy. The result is a numerical integration procedure applicable to stiff and nonstiff problems alike; whereas the classical integrator of Kutta is applicable only to nonstiff problems.

The paper begins with a discussion of the peripheral procedures that must accompany an integrator in order to construct a robust integration algorithm. This discussion includes a startup procedure, the ability to estimate error and to control it, and a scheme for recording the results. The startup procedure determines the initial step size by obtaining a Taylor series expansion of the solution in the neighborhood of the initial state. The error estimate used permits a ratio of absolute to relative error measures. A self-adaptive time stepper is used to control this error. The controller employed was derived from control theory by Gustafsson et al. (1988). It is a proportional integral controller with antiwindup; in other words, the controller has a proportional feedback mechanism for maintaining algorithm stability. This mechanism permits the user to obtain any density of output without impairing the efficiency of the integrator and its controller. The local, Hermite, interpolation procedure of Shampine (1985) is used.

An integrator must accompany these peripheral procedures to complete the algorithm. The next topic of discussion is the basic mathematical structure of explicit Runge-Kutta methods. Specifically, "the" classic Runge-Kutta integrator originally derived by Kutta (1901) is presented. A fifth stage is added to his method, producing a first-same-as-last (FSAL) integrator with an embedded solution. This addition is done to provide the error estimator with two approximate solutions of different accuracies and is designed so as to provide this information as efficiently as possible.

Theoretically, a Rosenbrock (1963) integrator is an extension to Runge-Kutta integrators, yet no Rosenbrock integrator known to the authors has had its coefficients derived with this fact in mind. Herein, we take those coefficients common to both Runge-Kutta and Rosenbrock integrators and fix them to the values of "the" classic integrator. The order conditions of Wolfbrandt (1977) are then applied to solve for the remaining coefficients unique to Rosenbrock integrators. The result is a semi-implicit integrator with superior capabilities. The derived formulæ can be encoded efficiently by introducing a transformation into the set of equations that constitute a Rosenbrock-Wolfbrandt integrator.

With a compatible pair of Runge-Kutta and Rosenbrock integrators, one can devise a scheme to switch from one to the other when the solution leaves a stiff domain and enters a nonstiff domain, or vice versa, thereby providing an algorithm with optimum efficiency. This scheme requires a cheap means of assessing whether the solution is stiff or not at its current position in state space. An algorithm is presented that attempts to accomplish this goal. It is not as rigorously based as one would like, but it does function properly.

Now that these integrators have been encapsulated into an algorithm for robust and efficient integration, it is time to test the product to determine its worth. To do this, five examples with various degrees of stiffness and/or stability are considered. All five have roots as mathematical descriptions of some physical process or another. To determine the worth of our integrators, we have compared them against two "commercial benchmarks." The first is the explicit 4(5) Runge-Kutta integrator of Fehlberg (1969). Here the intention is to make comparisons for problems that are not stiff. The second integrator is a 4(3) Rosenbrock integrator (like ours), derived by Shampine (1982) and advocated by Press et al. (1992) as the integrator of choice for stiff systems that are not too large (10 equations or fewer). These integrators were compared as fairly as we could, from a code point of

view, and almost without exception our Rosenbrock method was the superior integrator, even for problems that are not stiff.

The coefficients for the explicit and semi-implicit integrators are derived in appendix A.

Please contact the first author in writing as to the availability of software. The Windows/DOS, Sun/UNIX, and Oberon/F platforms are supported. Documentation is included.

Chapter 2

The Integrators

2.1 The Problem Statement: Defining the Algorithm

Many physical processes (e.g., chemical kinetics and inelastic material evolution) are represented by an initial value problem (IVP) associated with the solution of a set of ordinary differential equations (ODE's) of first order, as described by

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}\big(t, \mathbf{x}(t)\big) \,, \tag{2.1}$$

or equivalently by

$$\dot{\mathbf{x}} = \mathbf{f}\big(t, \mathbf{x}(t)\big) \,,$$

with a boundary condition for the initial value prescribed as

$$\mathbf{x}(t_0) = \mathbf{x}_0 \,,$$

where $\mathbf{x}$ is a vector of M dependent scalar variables $\{x^1 \;\; x^2 \;\; \ldots \;\; x^M\}^{\mathsf{T}}$ whose initial value at the time $t = t_0$ is $\mathbf{x}_0$. Time is taken as the independent variable for illustrative purposes. The overdot " $\dot{\cdot}$ " is used to denote differentiation with respect to time. If $\mathbf{f}$ does not depend *explicitly* on time t, the system is said to be autonomous; otherwise, it is considered nonautonomous. The reader interested in studying various methods for the numerical integration of ODE's is referred to the excellent texts of Hairer et al. (1991, 1993). We choose to present two fairly simple methods in the following sections.

When constructing an algorithm for the numerical integration of a differential equation, one must first discretize the known ordinary *differential* equation (ODE), which is defined at a point (e.g., t_n), into an ordinary *difference* equation (OΔE), which is defined over an interval (e.g., $[t_n, t_{n+1}]$ where $t_{n+1} = t_n + h$ with h being the size of the time step). We therefore consider the following IVP, that is,

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\boldsymbol{\mathcal{F}}\big(t_{n+\theta}, \mathbf{x}_{n+\theta}\big) \tag{2.2}$$

with

$$\mathbf{x}_{n=0} = \mathbf{x}_0 \,,$$

where $t_{n+\theta}$ is an intermediate point belonging to the interval $t_n \le t_{n+\theta} \le t_{n+1}$ such that $0 \le \theta \le 1$. In general, $\boldsymbol{\mathcal{F}}$ need not equal $\mathbf{f}$, or equivalently, $\dot{\mathbf{x}}$. The one-step OΔE method presented in equation(2.2) is often used in numerical integration algorithms to represent the IVP of the first-order ODE given in equation(2.1). The integrators presented in this paper are constructed as weighted sums of one-step integrators. We adopt the notation that $\mathbf{x}_n \equiv \mathbf{x}(t_n)$, $\mathbf{x}_{n+\theta} \equiv \mathbf{x}(t_{n+\theta})$, where $t_{n+\theta} = t_n + \theta h$ with $0 \le \theta \le 1$ and $\mathbf{x}_{n+1} \equiv \mathbf{x}(t_{n+1})$.

2.1.1 Solution Interpolation

In the pursuit of acquiring a solution to a system of differential equations over a time interval $[t_0, t]$, two objectives are typically sought. First, a set of times $\{t_0, t_a, t_b, \ldots, t\}$ is prescribed by the user, such that for each element in this set the associated solution vector is to be recorded, thereby resulting in the set $\{\mathbf{x}_0, \mathbf{x}_a, \mathbf{x}_b, \ldots, \mathbf{x}\}$. This prescription is usually made so that the solution can be graphically displayed. Second, the integrator itself determines another set of times $\{t_0, t_1, t_2, \ldots, t_n, \ldots, t\}$ whereby it advances the solution while maintaining a prescribed level of accuracy. A difficulty arises in that usually $\{t_0, t_a, t_b, \ldots, t\} \not\subset \{t_0, t_1, t_2, \ldots, t\}$. From the viewpoint of computational efficiency, it is advantageous to let the integrator propagate along the solution via its optimum set $\{t_0, t_1, t_2, \ldots, t\}$ and to use interpolation, where necessary, to secure the solution vectors $\{\mathbf{x}_0, \mathbf{x}_a, \mathbf{x}_b, \ldots, \mathbf{x}\}$ at the prescribed times $\{t_0, t_a, t_b, \ldots, t\}$.

Interpolation permits any ensemble of $\mathbf{x}$'s to be recorded without impairing the efficiency of the integrator by unnecessarily altering its time-step size. In this report, we present integration formulæ designed for use with a specific form of local interpolation. It is

assumed that the solutions $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$, along with their slopes $\mathbf{f}_n$ and $\mathbf{f}_{n+1}$, are known to the same order of accuracy (i.e., to $\mathcal{O}(h^{p+1})$) at the end of an accepted integration step. A cubic Hermite interpolant for $\mathbf{x}_{n+\theta}$ $(0 \le \theta \le 1)$, which is local to the interval $[t_n, t_{n+1}]$, is then given by (Shampine 1985):

$$\begin{aligned}\mathbf{x}_{n+\theta} = &(1-\theta)\mathbf{x}_n + \theta\mathbf{x}_{n+1} \\ &+ \theta(1-\theta)\big((1-2\theta)(\mathbf{x}_n - \mathbf{x}_{n+1}) \\ &\quad + (1-\theta)h\mathbf{f}_n - \theta h\mathbf{f}_{n+1}\big) \\ &+ \mathcal{O}(h^4) + \mathcal{O}(h^{p+1})\,.\end{aligned} \tag{2.3}$$

The $\mathcal{O}(h^4)$ error belongs to the interpolation formula, whereas the $\mathcal{O}(h^{p+1})$ error belongs to the integrator, and hence, to the values $\mathbf{x}_n$, $\mathbf{x}_{n+1}$, $\mathbf{f}_n$, and $\mathbf{f}_{n+1}$. Consequently, $\mathbf{x}_{n+\theta}$ can be no more than third-order accurate. Higher-order Hermite interpolation formulæ can be derived by using a formula given in Shampine (1985) (see also Hairer et al., 1993, pp. 188–195). Because values for both the function and its derivative on the right side of the time interval $[t_{n-1}, t_n]$, say, coincide exactly with those on the left side of the next time interval $[t_n, t_{n+1}]$, Hermite interpolation produces a global, continuous, and differentiable approximation to the solution.

2.1.2 Time-Step Control

For the OΔE integrator to be robust, it must maintain some adaptive control over its own progress, thereby making frequent changes to the size of its time step as it advances along the path of its solution. The objective of automatic time-step control is to achieve a stable solution of predetermined accuracy with minimal computational effort. To accomplish this objective, the integration algorithm must compute two approximations to the solution (viz., $\mathbf{x}_{n+1}$ and $\widehat{\mathbf{x}}_{n+1}$, each with a different order of accuracy). An error estimate is then constructed by taking their difference (i.e., $\mathbf{x}_{n+1} - \widehat{\mathbf{x}}_{n+1}$) and if sufficiently small, advancing $\mathbf{x}_{n+1}$ (the more accurate of the two results) as the accepted approximation to the actual solution at that point. An estimate for the error can then be acquired from the relationship

$$\epsilon_{n+1} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(\frac{x^m_{n+1} - \widehat{x}^m_{n+1}}{x^m_{\text{weight}(n+1)}}\right)^2}\,, \tag{2.4}$$

where

$$x^m_{\text{weight}(n+1)} = \mathfrak{f}_a x^m_{\text{range}} + \mathfrak{f}_r |x^m_{n+1}|\,.$$

Here x^m_{n+1} is the mth element, or dependent variable, of the M elements belonging to the solution vector $\mathbf{x}_{n+1}$ at the $n+1$ time step, with a similar explanation applying to $\widehat{x}^m_{n+1}$. The user should attempt to specify $x^m_{\text{range}} > 0$ for all m, which denotes an absolute measure for the maximum range that each dependent variable x^m is expected to vary over. These scaling factors should ideally be set to within an order of magnitude of expected values. If left unspecified, they will be set to 1 by default in our software. The fractions $\mathfrak{f}_a$ $(0 < \mathfrak{f}_a < 1)$ and $\mathfrak{f}_r$ $(\mathfrak{f}_r = 1 - \mathfrak{f}_a)$ partition the weighting factors $x^m_{\text{weight}(n+1)}$ between "absolute" and "relative" measures and are also user specified. A default value of 1/2 has been assigned to both $\mathfrak{f}_a$ and $\mathfrak{f}_r$ if they are unspecified.

An effective, automatic, time-stepping routine for numerical integration is given by (cf. Hairer and Wanner, 1991, pp. 31–35)

$$\frac{h_{\text{next}}}{h} = \begin{cases} \min\left[\mathfrak{g}_{\max},\ \left(\frac{\epsilon_{\max}}{\epsilon_{n+1}}\right)^{\frac{0.7}{p+1}}\left(\frac{\epsilon_n}{\epsilon_{\max}}\right)^{\frac{0.4}{p+1}}\right] \\ \qquad \text{if } \epsilon_{n+1} \le \epsilon_{\max}\,, \\ \max\left[\mathfrak{g}_{\min},\ \left(\frac{\epsilon_{\max}}{\epsilon_{n+1}}\right)^{\frac{1}{p}}\right] \\ \qquad \text{if } \epsilon_{n+1} > \epsilon_{\max}\,, \end{cases} \tag{2.5}$$

which, in the terminology of control theory, is a proportional integral controller with antiwindup (Gustafsson et al., 1988). Here $\epsilon_{\max} > 0$ is the maximum allowable error prescribed by the user (e.g., $\epsilon_{\max} = 10^{-p}$); ϵ_{n+1} is the error of the most recent integration increment, as determined from equation(2.4), with h being the size of its time step and h_{next} the projected size for the ensuing time step; ϵ_n is the error from the prior successful time step; and p is the order of the integrator. The exponents of the error ratios for a successful step have a denominator of p+1, whereas the exponent of the error ratio for an unsuccessful step has a denominator of p. The reason is that the error of integration has, at least, exceeded that of $\widehat{\mathbf{x}}_{n+1}$ for an unsuccessful step, which is taken to be of order $p-1$ in the methods presented herein, and therefore has an error of order p.

The additional term $(\epsilon_n/\epsilon_{\max})^{0.4/(p+1)}$ found in this construct is the "proportional feedback" that Gustafsson et al. (1988) added to the "control circuit" of this time stepper. This dampens the controller, thereby increasing its stability. In effect, it serves as a variable factor of safety. Therefore, an explicit factor of safety, which is common in many other time-stepping algorithms (cf. Hairer et al., 1993, p. 168), is not present in Gustafsson's approach.

If ϵ_{n+1} is larger than $\epsilon_{\max}$, the value determined for h_{next} will reflect a decrease in the step size *after we reject the present (failed) time step and retry the integration with the new estimate* h_{next} *for* h, which,

in the interest of algorithmic stability, is not permitted to be less than g_{min} times the size of the prior attempt. If ϵ_{n+1} is smaller than ϵ_{max}, on the other hand, the value determined for h_{next} will reflect a possible increase in the step size *to be applied to the next integration step*, which, in the interest of algorithmic stability, is not permitted to be greater than g_{max} times the size of the prior step. This procedure is antiwindup. Shampine and Watts (1979) suggest that g_{max} and g_{min} take on the values 5 and 1/10, respectively. In our code, these parameters are user definable with defaults as suggested by Shampine and Watts.

A separate procedure should to be used to determine the initial time-step size at startup. The procedure imposed in our software begins by evaluating $\mathbf{f}_0 = \mathbf{f}(t_0, \mathbf{x}_0)$ at the initial point. We then consider the norm

$$\|\mathbf{z}\|_\epsilon = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(\frac{z^m}{x^m_{\text{weight}(0)}}\right)^2}, \qquad (2.6)$$

where $\mathbf{z}$ is any vector with elements $\{z^1 \;\; z^2 \;\; \ldots \;\; z^M\}^T$ (in particular, $\mathbf{z}$ is either $\mathbf{x}$ or $\mathbf{f}$) and where $x^m_{\text{weight}(0)} = f_a x^m_{\text{range}} + f_r|x^m_0|$. The time-step size of

$$h_0 = \frac{\|\mathbf{x}_0\|_\epsilon}{\|\mathbf{f}_0\|_\epsilon} \qquad (2.7)$$

is assigned as our first guess for the beginning step size, such that h_0 is bound by the user-defined interval $[h_{min}, h_{max}]$. If $\|\mathbf{x}_0\| = 0$, then h_0 is set to h_{min}. Typical values given to h_{min} and h_{max} might be $t/1000$ and $t/10$, respectively, where t is the total time of integration. A selection for these values is based on the user's knowledge of his/her given problem.

A refined estimate for the initial time-step size is obtained by performing one explicit, Euler, integration step (i.e., $\mathbf{x}_1 = \mathbf{x}_0 + h_0\mathbf{f}_0$) and then evaluating $\mathbf{f}_1 = \mathbf{f}(t_0 + h_0, \mathbf{x}_1)$. Using the same norm as before, we assign

$$h = 2\left|\frac{\|\mathbf{x}_1\|_\epsilon - \|\mathbf{x}_0\|_\epsilon}{\|\mathbf{f}_1\|_\epsilon + \|\mathbf{f}_0\|_\epsilon}\right| \qquad (2.8)$$

to be the starting step size, with $h_{min} \leq h \leq h_{max}$. This equation comes from the second-order approximation $\|\mathbf{x}_1\|_\epsilon \simeq \|\mathbf{x}_0\|_\epsilon + h\|\mathbf{f}_0\|_\epsilon + \frac{1}{2}h^2[(\|\mathbf{f}_1\|_\epsilon - \|\mathbf{f}_0\|_\epsilon)/h]$, where $\|\dot{\mathbf{f}}_0\|_\epsilon \approx (\|\mathbf{f}_1\|_\epsilon - \|\mathbf{f}_0\|_\epsilon)/h$. This procedure, or one similar to it, will usually give a good estimate for the initial step size and thereby eliminate potentially wasted computing time that could otherwise be caused by poor initial estimates.

The methods of local interpolation, error estimation, and time-step control that have been outlined above apply to both the explicit and semi-implicit Runge-Kutta formulæ that follow. Collectively, they constitute the integration algorithm employed in our software. This algorithm is not too difficult to implement, and it can give good results for a variety of applications.

2.2 An Explicit Runge-Kutta Integrator

The appropriate selection of an integrator depends on one's application. Applications of interest to the authors involve solving systems of first-order ODE's that are coupled and extremely nonlinear. Under these conditions, higher-order Runge-Kutta formulæ can actually be more costly to execute than lower-order formulæ. As a compromise between accuracy and efficiency, a five-stage, fourth-order, Runge-Kutta integrator is considered.

Two approaches are used in securing the necessary error needed to control the step size of Runge-Kutta integrators. The first approach is *step doubling* (i.e., Richardson extrapolation) where each time step is integrated twice, once as a full step and then, independently, as two half-steps. This approach is a popular but expensive means for monitoring the error. The preferred approach is *embedded integration* (i.e., a Fehlberg method) where Runge-Kutta formulæ of order p have embedded in them another Runge-Kutta integrator of order $p-1$, say. This method has not been as popular as the first, but it is much more efficient.

The following embedded Runge-Kutta integrator was designed to permit local interpolation by using equation(2.3) in order to obtain any $\mathbf{x}_{n+\theta}$ found within the interval $[t_n, t_{n+1}]$ for the purpose of securing dense output. This procedure allows for optimum time stepping by, in effect, separating the process of integration from the process of data collection.

Explicit Runge-Kutta methods are applicable to situations where the system of ODE's is *stable* and *not stiff*. By stable, we mean that the real part of all eigenvalues belonging to the Jacobian are negative valued (i.e., $\Re(\lambda_i) < 0 \; \forall \, i$) and that the product of the time-step size and the maximum eigenvalue is bound (viz., $h \max_i \Re(|\lambda_i|) < c$, where c is a constant whose value is usually of order 1). (It is actually integrator dependent.) This requirement implies that the system has built-in dampening. By nonstiff, we mean that the real part in the ratio of the maximum to minimum eigenvalues of the Jacobian is of order

1 (i.e., $\max_i \Re(|\lambda_i|)/\min_j \Re(|\lambda_j|) = \mathcal{O}(1)$). Nonstiffness implies that the rate of change in each dependent variable is roughly the same.

On the topic of stiffness, Shampine and Watts (1979) state that explicit "Runge-Kutta methods are not suitable for a problem which is *very* stiff, although tests show that among methods for nonstiff problems, they are likely to be the best in the presence of mild stiffness." A method more appropriate for stiff integration is presented in the next section.

The explicit Runge-Kutta integrator developed in appendix A is of the Fehlberg (1969) type and is described by the formulæ

$$\left.\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + h\sum_{i=0}^{4} c_i\mathbf{k}_i + \mathcal{O}(h^5) \\ \widehat{\mathbf{x}}_{n+1} &= \mathbf{x}_n + h\sum_{i=0}^{4} \widehat{c}_i\mathbf{k}_i + \mathcal{O}(h^4) \end{aligned}\right\} \tag{2.9}$$

with an error estimate of

$$\epsilon_{n+1} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left[\frac{h\sum_{i=0}^{4}(c_i-\widehat{c}_i)k_i^m}{x^m_{\text{weight}(n+1)}}\right]^2}, \tag{2.10}$$

where k_i^m is the mth element of derivative vector $\mathbf{k}_i = \{k_i^1 \;\; k_i^2 \;\; \ldots \;\; k_i^M\}^{\mathsf{T}}$. These Runge-Kutta derivatives are evaluated as

$$\mathbf{k}_i = \mathbf{f}\big(t_n + a_i h, \mathbf{x}_n + h\sum_{j=0}^{i-1} a_{ij}\mathbf{k}_j\big)\,, \tag{2.11}$$

given the FSAL constraints

$$\left.\begin{aligned} \mathbf{k}_{0(n+1)} &= \mathbf{f}(t_n, \mathbf{x}_n) \equiv \mathbf{k}_{4(n)} \\ \mathbf{k}_{4(n+1)} &= \mathbf{f}\big(t_n + h, \mathbf{x}_n + h\sum_{j=0}^{3} c_j\mathbf{k}_j\big) \\ &\equiv \mathbf{f}(t_{n+1}, \mathbf{x}_{n+1}) = \mathbf{k}_{0(n+2)} \end{aligned}\right\} \tag{2.12}$$

and therefore $c_4 = 0$ in equation(2.9).

As a result of the constraints implied in equation(2.12), the number of derivative evaluations is reduced from $p+1$ (5) to p (4) for each successful time step, because $\mathbf{k}_{0(n+1)} \equiv \mathbf{k}_{4(n)}$, and therefore $\mathbf{f}_n = \mathbf{k}_{4(n)}$. This property is characteristic of FSAL integrators. At startup, $\mathbf{k}_0$ must be explicitly evaluated. (Actually, it is precalculated in the startup procedure defined by equations(2.7) and (2.8).)

The solution is advanced at the completion of a successful step by assigning $\mathbf{x}(t_n + h) \Leftarrow \mathbf{x}_{n+1}$ and $\mathbf{f}(t_{n+1}, \mathbf{x}_{n+1}) \Leftarrow \mathbf{f}_{n+1} = \mathbf{k}_{4(n+1)}$, where $\mathbf{x}(t_n) \Leftarrow \mathbf{x}_n$ and $\mathbf{f}(t_n, \mathbf{x}_n) \Leftarrow \mathbf{f}_n = \mathbf{k}_{4(n)}$ were assigned in the prior step.

TABLE 2.1: COEFFICIENTS OF KUTTA'S CLASSIC INTEGRATOR WITH FIFTH-STAGE ADDED TO PRODUCE EMBEDDED 4(3) RUNGE-KUTTA INTEGRATOR

i	a_i	a_{ij}				
0	0	0				
1	1/2	1/2	0			
2	1/2	0	1/2	0		
3	1	0	0	1	0	
4	1	1/6	1/3	1/3	1/6	0

i	c_i	$\widehat{c}_i$	$c_i - \widehat{c}_i$
0	1/6	1/6	0
1	1/3	1/3	0
2	1/3	1/3	0
3	1/6	0	1/6
4	0	1/6	−1/6

Efficiency of integration comes from the fact that the functions $\mathbf{k}_0, \mathbf{k}_1, \ldots, \mathbf{k}_p$, which are the *unknowns*, are the same for both formula $\mathbf{x}_{n+1}$ and formula $\widehat{\mathbf{x}}_{n+1}$, which are of orders p and $p-1$, respectively. What distinguishes $\mathbf{x}_{n+1}$ from $\widehat{\mathbf{x}}_{n+1}$ are their respective sets of *known* coefficients (i.e., $\{c_0, c_1, \ldots, c_p\}$ and $\{\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_p\}$). This distinction is why embedded Runge-Kutta methods (i.e., Fehlberg methods) are so efficient. Such a set of coefficients for a 4(3)-order integrator belonging to the class of integrators described by equations(2.9) to (2.12) is given in table 2.1. The first number in the pair $p(p–1)$ represents the order of the base solution $\mathbf{x}$, with the second bracketed number giving the order of the embedded solution $\widehat{\mathbf{x}}$. The derivation of the coefficients presented in this table is given in appendix A.

The base integrator for $\mathbf{x}$ (i.e., stages 0 to 3 presented in table 2.1) is the classic Runge-Kutta method originally derived by Kutta (1901, p. 443). His integrator has a maximum truncation error of 1/80 at a fourth order of accuracy (cf. Fehlberg, 1969). The coefficients $\widehat{c}_i$ belonging to the embedded solution $\widehat{\mathbf{x}}$ were obtained by using equation(4.9) of Hairer et al. (1993, p. 167). The embedded integrator has a maximum truncation error of 1/72 at a $p-1$ (third) order of accuracy.

2.3 A Semi-Implicit Runge-Kutta Integrator

The explicit Runge-Kutta method presented in the previous section works well for the numerical integration of systems of first-order ODEs that are not stiff, that are not unstable, or whose Jacobians are

not singular. However, when stiffness, instability, or Jacobian singularity become an issue, formulæ like the semi-implicit Runge-Kutta method that we now present should be used instead.

A system of ODE's is *unstable* if the real part of at least one eigenvalue belonging to the Jacobian is positive (i.e., $\Re(\lambda_i) > 0$ for one or some i). The system is said to be *singular* if at least one eigenvalue is zero valued (i.e., the determinant of the Jacobian is singular). Stiffness has to do with the ratio of eigenvalues and therefore requires at least two differential equations. Stiffness does not have a universally accepted definition. As we use the term, a system is *stiff* if $\max_i \Re(|\lambda_i|)/\min_j \Re(|\lambda_j|) \gg 1$, with the value of 100 representing a "fuzzy boundary" between not being stiff and being stiff. Stiffness implies that one dependent variable evolves rapidly while another evolves slowly.

Rosenbrock (1963) proposed a class of Runge-Kutta integrators that are semi-implicit and therefore of potential use for stiff integration. Wolfbrandt (1977) modified Rosenbrock's method, resulting in a more efficient means for its computation. Using the order conditions derived by Wolfbrandt (see also, Kaps and Wanner, 1981), we have constructed an embedded, semi-implicit, Runge-Kutta integrator (i.e., a Rosenbrock method) of order 4 that contains the explicit Runge-Kutta integrator of table 2.1 as a subset. Therefore, a software implementation can transition easily and systematically from the semi-implicit integrator to its explicit form, or vice versa, at any point in the integration where the system of ODE's either leaves or enters, respectively, a domain of stiffness.

The semi-implicit Runge-Kutta integrator considered is described by the formulæ

$$\left.\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + h\sum_{i=0}^{4} c_i \mathbf{k}_i + \mathcal{O}(h^5) \\ \widehat{\mathbf{x}}_{n+1} &= \mathbf{x}_n + h\sum_{i=0}^{4} \widehat{c}_i \mathbf{k}_i + \mathcal{O}(h^4) \end{aligned}\right\} \qquad (2.13)$$

with an error estimate of

$$\epsilon_{n+1} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left[\frac{h\sum_{i=0}^{4}(c_i - \widehat{c}_i)k_i^m}{x^m_{\text{weight}(n+1)}}\right]^2} \qquad (2.14)$$

What distinguishes this semi-implicit integrator from the prior explicit one is the calculation of the Runge-Kutta derivatives, which are evaluated by

$$\mathbf{k}_i = \underbrace{\mathbf{f}\left(t_n + a_i h, \mathbf{x}_n + h\sum_{j=0}^{i-1} a_{ij}\mathbf{k}_j\right)}_{\text{explicit part}} + \underbrace{h\, b_i \frac{\partial \mathbf{f}_n}{\partial t} + h\frac{\partial \mathbf{f}_n}{\partial \mathbf{x}}\cdot\sum_{j=0}^{i} b_{ij}\mathbf{k}_j}_{\text{semi-implicit part}} . \qquad (2.15)$$

Equations (2.13) and (2.14) are identical to their respective explicit counterparts given in equations(2.9) and (2.10). Equations (2.11) and (2.15), however, are significantly different. The expression for the actual derivative (i.e., $\mathbf{f}$) found in the relation for $\mathbf{k}_i$ given in equation(2.15) has the same form as that which is found in equation(2.11). It is the Jacobian, $\partial \mathbf{f}_n/\partial \mathbf{x}$, and the nonautonomous gradient, $\partial \mathbf{f}_n/\partial t$, that provide the additional contributions to a Rosenbrock integrator.

Notice that $\partial \mathbf{f}_n/\partial \mathbf{x}$ and $\partial \mathbf{f}_n/\partial t$ are both evaluated at the beginning of the time step and are therefore evaluated *explicitly*—only once per time step. For this reason, Rosenbrock integration has been reported *not* to produce good results if approximate numerical evaluations are used to acquire $\partial \mathbf{f}_n/\partial \mathbf{x}$ and $\partial \mathbf{f}_n/\partial t$ (Shampine, 1982). *Analytic derivatives are preferred to ensure good quality results from Rosenbrock integrators.* Approximate numerical evaluations are less desirable, in part, because the Jacobian belongs to the integrator; whereas in other implicit integration algorithms, such as backward Euler, the Jacobian does not belong to the integrator but rather to the solver (viz., Newton-Raphson iteration). Numerically acquired Jacobians also cost more to evaluate than analytic Jacobians.

For convenience of implementation, Kaps and Rentrop (1979) and Kaps et al. (1985) have successfully used finite differences to construct the Jacobian instead of analytic derivation. In their 1985 paper they write the following: "It should be noted that this has two disadvantages: First, the numerical computation needs more computing time than an analytic version, because n function evaluations are necessary for one numerical Jacobian. Second, the order conditions were derived under the assumption of an exact Jacobian. If the truncation or rounding errors in the numerical computation of the Jacobian are not negligible the method may lose its order. However, this effect was seen in our computation only for very sensitive problems."

Insisting on "analytic" evaluations for the derivatives is sometimes perceived as being a shortcom-

TABLE 2.2: COEFFICIENTS OF OUR 4(3) ROSENBROCK METHOD ($b = 1/4$)

i	a_i	a_{ij}				
0	0	0				
1	1/2	1/2	0			
2	1/2	0	1/2	0		
3	1	0	0	1	0	
4	1	1/6	1/3	1/3	1/6	0

i	b_i	b_{ij}				
0	1/4	1/4				
1	0	−1/4	1/4			
2	−1/8	1/4	−5/8	1/4		
3	0	1/2	−3/4	0	1/4	
4	0	0	−1	1	−1/4	1/4

i	c_i	$\widehat{c}_i$	$c_i - \widehat{c}_i$
0	1/6	1/6	0
1	1/3	1/3	0
2	1/3	1/3	0
3	1/6	0	1/6
4	0	1/6	−1/6

ing of Rosenbrock methods, but if adhered to, the added effort required to derive a Jacobian is greatly rewarded by the enhanced computational efficiency that it brings to these integrators. Symbolic mathematical packages make securing an analytic Jacobian not as taxing as it once was.

Rosenbrock's (1963) original formulæ require an "explicit" Jacobian to be evaluated *for each stage* of integration, resulting in a semi-implicit method. Like implicit methods, semi-implicit methods require the solution of a linear system of equations; however, unlike implicit methods, they do not require the added burden of iteration to accomplish the task of solving the system. Wolfbrandt (1978, p. 90) generalized Rosenbrock's method by considering a fully populated lower-triangular matrix of coefficients for the b_{ij}, as indicated in equation(2.15). In addition, by constraining the diagonal terms to all be equal (i.e., $b_{ii} = b \ \forall \ i$), thereby utilizing a single explicit Jacobian, Wolfbrandt substantially streamlined the method, since now only one matrix inversion is required for each integration step instead of the p+1 separate matrix inversions originally proposed by Rosenbrock. The constants of our 4(3) Rosenbrock integrator are given in table 2.2. Their motivation and derivation are provided in appendix A.

The number of function evaluations (viz., $\mathbf{f}$) reduce from $p+1$ (5) to p (4) for each successful time step of this FSAL integrator. The reason is that $\mathbf{f}_n$ is contained within $\mathbf{k}_0$, whereas $\mathbf{f}_{n+1}$ is contained within $\mathbf{k}_4$ and can therefore be reused in evaluating $\mathbf{k}_0$ in the next step. These are also the derivatives found in the formula for local interpolation given in equation(2.3).

The base integrator for $\mathbf{x}$ given in table 2.2 has a maximum truncation error of 1/60 at a fourth order of accuracy, whereas the embedded integrator for $\widehat{\mathbf{x}}$ has a maximum truncation error of 1/48 at a third order of accuracy. These errors are slightly larger than, yet comparable to, the truncation errors of the explicit Runge-Kutta integrator given in table 2.1.

2.3.1 Code Implementation

A direct implementation of equations(2.13) to (2.15) requires, at each stage, the solution of a linear system of equations by using the matrix $[\frac{1}{b}\mathbf{I} - h\mathbf{J}]$, where $\mathbf{J} = (\partial \mathbf{f}_n / \partial \mathbf{x})$ is the Jacobian. In addition, it also requires the matrix-vector multiplication $h(\partial \mathbf{f}_n / \partial \mathbf{x}) \cdot \sum_{j=0}^{i-1} b_{ij} \mathbf{k}_j$, which fortunately can be avoided by introducing a change in variable (Hairer and Wanner, 1991, pp. 120–121), that is,

$$\boldsymbol{\kappa}_i = \sum_{j=0}^{i} b_{ij} \mathbf{k}_j \ , \quad i = 0, 1, \ldots, 4 \ . \tag{2.16}$$

This change leads to a more efficient implementation of Rosenbrock methods by removing the Jacobian from the right-hand side of the system being solved (Wolfbrandt, 1978, pp. 106–107). Because $b_{ii} = b \neq 0 \ \forall \ i$, the matrix b_{ij} has an inverse b_{ij}^{-1}. Consequently, the $\mathbf{k}_i$ can be recovered from the $\boldsymbol{\kappa}_i$ by the relation

$$\mathbf{k}_i = \sum_{j=0}^{i} b_{ij}^{-1} \boldsymbol{\kappa}_j \equiv \frac{1}{b} \boldsymbol{\kappa}_i - \sum_{j=0}^{i-1} \beta_{ij} \boldsymbol{\kappa}_j \ , \tag{2.17}$$

where

$$\beta_{ij} = \operatorname{diag}\left[\tfrac{1}{b} \quad \tfrac{1}{b} \quad \ldots \quad \tfrac{1}{b}\right] - b_{ij}^{-1}$$

is a lower-triangular matrix with zeros along the main diagonal.

Inserting the transformation of equation(2.17) into the Rosenbrock method of equations(2.13) and (2.15) and scaling the latter by dividing through by h lead to a more efficient implementation by the formulæ

$$\left.\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n + h \sum_{i=0}^{4} \chi_i \boldsymbol{\kappa}_i + \mathcal{O}(h^5) \\ \widehat{\mathbf{x}}_{n+1} &= \mathbf{x}_n + h \sum_{i=0}^{4} \widehat{\chi}_i \boldsymbol{\kappa}_i + \mathcal{O}(h^4) \end{aligned}\right\} \tag{2.18}$$

with error estimate

$$\epsilon_{n+1} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left[\frac{h\sum_{i=0}^{4}(\chi_i - \widehat{\chi}_i)\kappa_i^m}{x^m_{\text{weight}(n+1)}}\right]^2}, \quad (2.19)$$

where κ_i^m is the mth element of the transform vector $\boldsymbol{\kappa}_i = \{\kappa_i^1 \quad \kappa_i^2 \quad \ldots \quad \kappa_i^M\}^{\mathsf{T}}$. These Runge-Kutta-like derivatives are evaluated by solving, at each stage, the linear system

$$\begin{aligned}\left(\frac{1}{b}\mathbf{I} - h\frac{\partial \mathbf{f}_n}{\partial \mathbf{x}}\right)\boldsymbol{\kappa}_i \\ = \mathbf{f}\left(t_n + a_i h, \mathbf{x}_n + h\sum_{j=0}^{i-1}\alpha_{ij}\boldsymbol{\kappa}_j\right) \\ + h\,b_i\frac{\partial \mathbf{f}_n}{\partial t} + \sum_{j=0}^{i-1}\beta_{ij}\boldsymbol{\kappa}_j\,,\end{aligned} \quad (2.20)$$

wherein the above equations employ the following changes in variables:

$$\left.\begin{aligned}\alpha_{ij} &= \sum_{k=0}^{4} a_{ik}b_{kj}^{-1} \\ \chi_i &= \sum_{j=0}^{4} c_j b_{ji}^{-1} \\ \widehat{\chi}_i &= \sum_{j=0}^{4} \widehat{c}_j b_{ji}^{-1}\end{aligned}\right\}. \quad (2.21)$$

The coefficients associated with the a_{ij}, b_{ij}, c_i, and $\widehat{c}_i$ of table 2.2 are given in table 2.3 for this streamlined implementation of our Rosenbrock method.

Equation (2.20) is a linear system of equations having the form $\mathbf{A}\cdot\mathbf{x}_i = \mathbf{b}_i$, whose solutions, $\mathbf{x}_i = \mathbf{A}^{-1}\cdot\mathbf{b}_i$, require a single matrix inversion $\mathbf{A}^{-1}$ applied to the multiple right-hand vectors $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_p$. Each right-hand vector can only be evaluated when its predecessors are known (e.g., solving for $\mathbf{b}_2$ requires knowledge of both $\mathbf{x}_0$ and $\mathbf{x}_1$). The method of lower-upper (LU) factorization is ideally suited for solving linear systems with multiple right-hand sides.

For a given step size h, the matrix $\mathbf{A} = \frac{1}{b}\mathbf{I} - h\mathbf{J}$ is a constant matrix, since the Jacobian $\mathbf{J}$ is evaluated at the beginning of the currently solved step by using the last converged solution. Matrix $\mathbf{J}$ may be singular or ill conditioned due to stiffness, with very extreme values for its norm. Whatever its condition is, as long as the whole matrix $[\frac{1}{b}\mathbf{I} - h\mathbf{J}]$ is not singular, the LU decomposition procedure can take care of these difficulties by appropriately scaling the matrix through a full pivoting process. The only feature that must be ensured is the existence of the inverse $[\frac{1}{b}\mathbf{I} - h\mathbf{J}]^{-1}$, in which case a unique solution vector is ensured. Whenever the coefficient matrix happens to be singular, this condition can easily be corrected by altering the step size h.

TABLE 2.3: COEFFICIENTS USED IN IMPLEMENTATION OF OUR 4(3) ROSENBROCK METHOD

i	a_i	α_{ij}				
0	0	0				
1	1/2	2	0			
2	1/2	2	2	0		
3	1	6	10	4	0	
4	1	14/3	20/3	4/3	2/3	0

i	b_i	β_{ij}				
0	1/4	0				
1	0	−4	0			
2	−1/8	−6	−10	0		
3	0	−4	−12	0	0	
4	0	4	12	16	−4	0

i	χ_i	$\widehat{\chi}_i$	$\chi_i - \widehat{\chi}_i$
0	14/3	10/3	4/3
1	20/3	8/3	4
2	4/3	−4/3	8/3
3	2/3	2/3	0
4	0	2/3	−2/3

The solution $\boldsymbol{\kappa}_i$ is plugged into the right-hand side vector of equation(2.20). As the coefficient matrix is constant, this vector essentially controls the solution process. Therefore, monitoring its behavior can detect stiffness or ill conditioning of the system.

2.3.2 Stiffness Detection

Whenever a nonstiff integrator encounters stiffness, the product of its step size with the dominant eigenvalue of the Jacobian will typically lie near the boundary of its stability domain. To avoid wasting too much effort when encountering stiffness with a nonstiff integrator or, in the opposite situation, to avoid wasting too much effort when accuracy (rather than stability) controls the time-step size in a stiff integrator, it is important that a code be equipped with a cheap means of detecting stiffness so that it can switch, whenever appropriate, between the two integrators.

A stiffness-checking algorithm that is both rigorous and inexpensive does not exist, to the best of our knowledge, for determining when to transition from a nonstiff to a stiff integrator. Instead, in our implementation, we use a simple rule of thumb. If the time-step size has been reduced three times with-

out successfully advancing the solution when using the explicit integrator, the Jacobian is evaluated and the method of integration switches over to the semi-implicit algorithm. A more mathematically pleasing criterion exists for detecting when to switch from a stiff to a nonstiff integrator.

To estimate directly the dominant eigenvalue, say $\lambda_{M(n)}$, of the Jacobian $\mathbf{J}_n = \partial \mathbf{f}_n / \partial \mathbf{x}$ from the nth time step, let $\mathbf{v}_n^{(M)}$ denote an approximation to its corresponding eigenvector such that $\|\mathbf{v}_n^{(M)}\|_2 \ll \|\mathbf{x}_n\|_2$, where the $\|\cdot\|_2$ is the standard L_2 vector norm. Then, by the mean value theorem,

$$|\lambda_{M(n)}| \approx \frac{\|\mathbf{f}(t_n, \mathbf{x}_n + \mathbf{v}_n^{(M)}) - \mathbf{f}(t_n, \mathbf{x}_n)\|_2}{\|\mathbf{v}_n^{(M)}\|_2} \tag{2.22}$$

will provide a good approximation to this leading eigenvalue (Hairer and Wanner, 1991, pp. 23–24). Because the Jacobian is explicit, this determination for the principal eigenvalue is also explicit in the sense that $\lambda_{M(n)}$ is associated with the beginning of the interval $[t_n, t_{n+1}]$. The calculation of $\lambda_{M(n)}$ will, in actuality, take place at the end of the previous time step over the interval $[t_{n-1}, t_n]$. The result of this evaluation determines if the current interval $[t_n, t_{n+1}]$ will be integrated with the Fehlberg method of table 2.1 or with the Rosenbrock method of table 2.2.

Because we have set $a_3 = a_4 = 1$, a natural choice in evaluating equation(2.22) for the Rosenbrock integrator described in table 2.2 is to take

$$|\lambda_{M(n+1)}| \approx \frac{\|\mathbf{k}_4 - \mathbf{k}_3\|_2}{\|\mathbf{x}_{n+1} - (\mathbf{x}_n + h\,\mathbf{k}_2)\|_2}\,, \tag{2.23}$$

or equivalently, to consider

$$h_{n+1}\,|\lambda_{M(n+1)}| \approx \frac{6\|\mathbf{k}_4 - \mathbf{k}_3\|_2}{\|\mathbf{k}_0 + 2\mathbf{k}_1 - 4\mathbf{k}_2 + \mathbf{k}_3\|_2}\,,$$

TABLE 2.4: INVERSE OF b_{ij} GIVEN IN TABLE 2.2

i	b_{ij}^{-1}				
0	4	0	0	0	0
1	4	4	0	0	0
2	6	10	4	0	0
3	4	12	0	4	0
4	−4	−12	−16	4	4

where $\mathbf{k}_4 = \mathbf{f}(t_{n+1}, \mathbf{x}_{n+1})$ and $\mathbf{k}_3 = \mathbf{f}(t_{n+1}, \mathbf{x}_n + h\,\mathbf{k}_2)$, with $\mathbf{x}_{n+1} - (\mathbf{x}_n + h\,\mathbf{k}_2)$ providing a good approximation to the eigenvector $\mathbf{v}_{n+1}^{(M)}$ corresponding to the dominant eigenvalue $\lambda_{M(n+1)}$, such that one obtains $\|\mathbf{x}_{n+1} - (\mathbf{x}_n + h\,\mathbf{k}_2)\|_2 \ll \|\mathbf{x}_{n+1}\|_2$. In the case of a Wolfbrandt-modified Rosenbrock integrator, one must transform the $\boldsymbol{\kappa}_i$ back to the $\mathbf{k}_i$ by the relation $\mathbf{k}_i = \sum_{j=0}^{i} b_{ij}^{-1} \boldsymbol{\kappa}_j$ (eq. (2.17)), where the matrix coefficients b_{ij}^{-1} pertaining to table 2.2 are given in table 2.4.

After an integration over the time increment $[t_{n-1}, t_n]$ has been determined to be successful while using the Rosenbrock integrator, but before advancing to the next time step, our code evaluates $h_{n+1}|\lambda_{M(n)}|$, wherein h_{n+1} is the next time step h_{next} coming from equation(2.5), which is proposed for the next time increment $[t_n, t_{n+1}]$. A decision is made according to

$$\text{if } h_{n+1}|\lambda_{M(n)}| \begin{cases} < 1 & \text{switch to Runge-Kutta} \\ & \text{integration for } [t_n, t_{n+1}], \\ \geq 1 & \text{continue with Rosenbrock} \\ & \text{integration for } [t_n, t_{n+1}]. \end{cases}$$

Chapter 3

Numerical Examples

In this chapter, a few examples are presented to verify our numerical method and to compare it with other, well-known, Runge-Kutta integrators. To this end, a variety of examples have been chosen from several scientific fields. Despite their different realms, all the examples have one common denominator: they are mathematically defined by a system of first-order, nonlinear, differential equations that present, to varying degrees, regional stiffness or instability.

Three Runge-Kutta-based integrators were used in the comparisons:

1. Fehlberg's (1969) 4(5) *explicit* algorithm. For ease of writing, it is abbreviated hereafter to EFA (explicit Fehlberg algorithm). This is a "commercial" Runge-Kutta integrator.

2. *Semi-implicit* Rosenbrock (1963) formulation with Shampine's (1982) 4(3) coefficients and an adjusting step-size scheme proposed by Kaps and Rentrop (1979). The whole package is referred to as SKA (Shampine-Kaps algorithm). This integrator is advocated by Press et al. (1992) for solving stiff ODE's.

3. *Semi-implicit* Rosenbrock (1963) scheme with the coefficients given in table 2.3 and the stepping procedure proposed by Gustafsson et al. (1988), equation(2.5). This algorithm is abbreviated FGA (Freed-Gustafsson algorithm).

The numerical features presented by the examples are the accuracy of the results, the total number of steps required for integration, and the residual error obtained during the solution process.

The first example was taken from the field of chemical kinetics (Hairer et al., 1993, pp. 115–116) and yields a two-dimensional periodic solution known as the Brusselator (i.e., it has a single attractor). The Brusselator, defined by an unstable and stiff (coefficient dependent) set of equations, served as the initial test case for verifying the derivation and implementation of our algorithm and for testing its internal interpolation scheme (eq. (2.3)), which transfers the solution from the algorithmic time stepper to an output time sequence required by the user.

In the second example, the two semi-implicit integrators were applied to Lorenz's (1963) classical chaos problem, which originated from simulating weather patterns in forecasting (see also Hairer et al., 1993, pp. 120–125). The result was a three-dimensional chaotic solution with two "strange attractors." The main characteristic here is the not quite periodic nature of the solution, leading to an eventual loss of determinism, hence, chaos. Unlike the first example, this system is defined by a nonstiff set of equations, and therefore its numerical solution is primarily an accuracy-driven process.

In the third example, the integration algorithms must deal with a 3×3 system of differential equations where one ODE is a linear combination of the other two ODE's and hence the system is singular. Its solution is asymptotic, rather than being periodic like the prior two examples. This system is test case D4 of Enright and Pryce (1987). Like the Brusselator, its origin is in chemical kinetics.

The fourth example is a viscoplastic material model (a material that has characteristics of both a plastic solid and a viscous fluid) developed by Robinson (1978)—a 13-degree-of-freedom system of ODE's. The intent is to describe behavior of metal structures operating at elevated temperatures. Like the prior example, the solution is asymptotic, provided that the direction of loading does not change. When change occurs, the solution goes through a transient on its way to a different asymptote. These transient domains are not stiff and therefore are accuracy driven. However, in the neighborhood of an asymptotic response the system becomes quite stiff, and consequently solutions are stability driven. An exclusive feature of this particular example is the presence of a step function in the evolution equations—"a source of numerical difficulties that must be dealt with" (Robinson and Swindeman, 1982).

Our final example is a four-degree-of-freedom system (i.e., test case F1 of Enright and Pryce (1987), which they state also has its source in chemical kinetics). This system is stable but *extremely* stiff (the stiffest in their library of IVP's), which is why it makes a good test case.

3.1 The Brusselator

In this first example, a set of equations defines a chemical evolution process known as the Brusselator. A more detailed discussion of its physical aspects is given in Hairer et al. (1993, pp. 115–116). In essence, this system of differential equations describes reactions involving several chemical substances. The more substances, the larger the system, in general. The cause of the Brusselator's numerical difficulties is directly related to the relative speed by which the concentrations of these chemicals change. The main features of the Brusselator include an unstable system and the existence of a limit cycle for the process.

The Brusselator describes the chemistry of six substances whose evolution through time is characterized by a stiff system of two differential equations in two unknowns, namely,

$$\left.\begin{aligned} \dot{y}_1 &= A + y_1^2 y_2 - (B+1)y_1 \\ \dot{y}_2 &= B y_1 - y_1^2 y_2 \end{aligned}\right\} \qquad (3.1)$$

with a steady state at $\mathbf{y} = \{A \quad B/A\}^{\mathsf{T}}$, where A and B are positive constants. The eigenvalues at this state are given by

$$\lambda = \frac{1}{2}\left[-(1 - B + A^2) \pm \sqrt{(1 - B + A^2)^2 - 4A^2}\right] .$$

Obviously, for $B > 1 + A^2$ the system is unstable. It exhibits "a limit cycle which, by numerical calculations, is seen to be unique" (Hairer et al., 1993, p. 116). To confirm this statement by using the FGA integrator, a few analyses were performed with $A = 1$ and $B = 3$ while changing the initial conditions. The final time was set at 20 sec, and the global tolerance for accuracy of integration was placed at 1×10^{-4}. The results are shown in figure3.1. Indeed, all solutions, regardless of the starting point, converge to one limit cycle. The only difference is the number of steps that it takes for them to get there. For the initial condition of $y_1 = 1.5$ and $y_2 = 3$, our analysis matches that of Hairer et al., with the exception that the integrators are different.

The solution and residual error, given as functions of time, are plotted in figure3.2. Comparing the results of this figure with those reported in the literature, we conclude that FGA provides accurate numbers with an error equal to or smaller than the preassigned tolerance. The analysis took 106 steps, rejecting 20 during the process. For the sake of comparison, the total number of steps required for the *explicit* integrator EFA was 842, while for the semi-implicit integrator SKA it was 2250, all at the same prescribed error tolerance and initial condition.

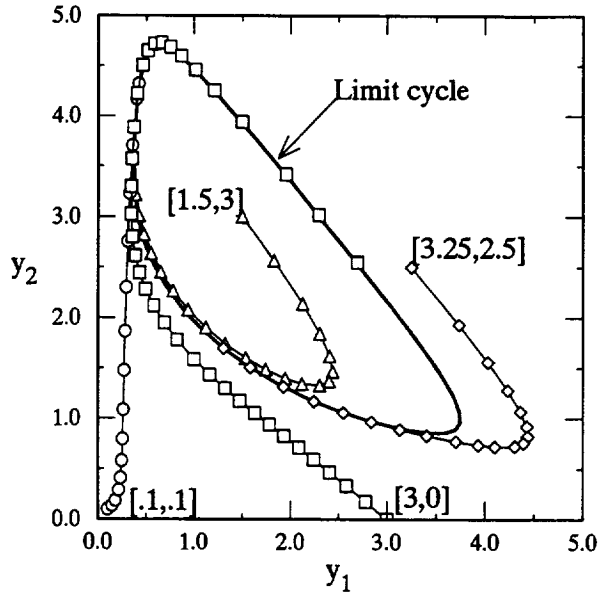
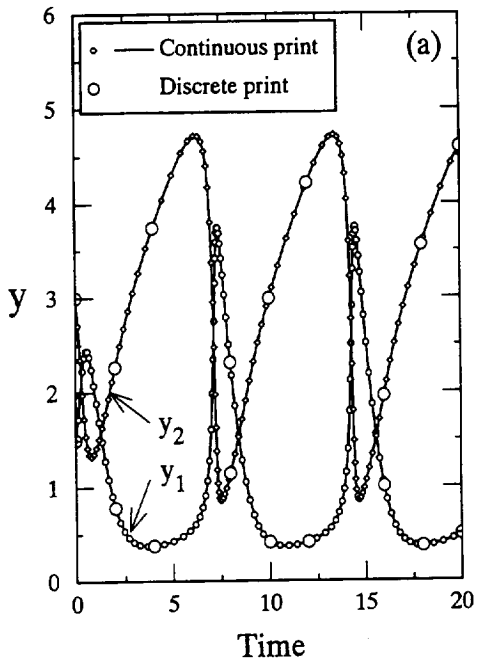


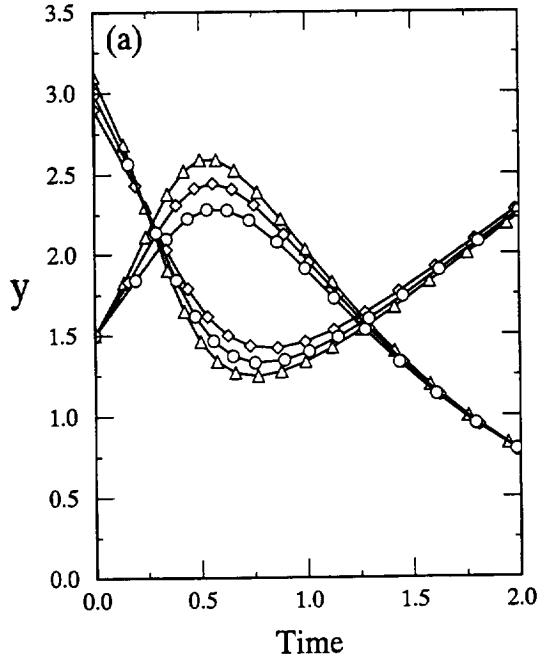
Figure 3.1: Chemical reaction: the limit cycle predicted by the Brusselator, as acquired by the semi-implicit integrator FGA.

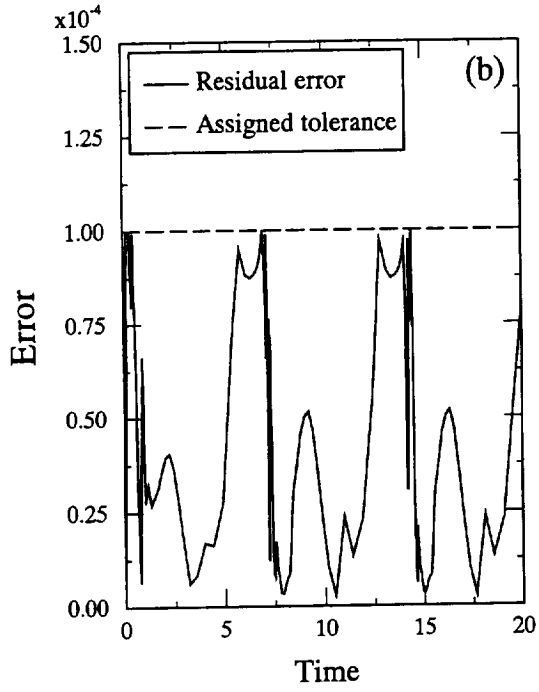
The interpolating routine provided values (labeled *discrete print*) on the continuous curve (fig. 3.2(a)) at the very points that were defined in the input file. Therefore, the time stepping of the algorithm is totally independent of the required output.

As a preface to our second example, a small perturbation was imposed on the initial conditions; the analysis was re-performed for three slightly different initial conditions (viz., $\{1.5 \quad 2.9\}^{\mathsf{T}}$, $\{1.5 \quad 3.0\}^{\mathsf{T}}$, and $\{1.5 \quad 3.1\}^{\mathsf{T}}$), and the final time was extended to 100 sec. The results are plotted in figure3.3. Despite the perturbations and the extreme time length of the analysis, after a transient period all solutions converged to the unperturbed solution—further indication that the limit cycle is unique.

After convincing ourselves of the correctness of our algorithm, we then approached the stiffness issue. By making B much bigger than A, thereby increasing the







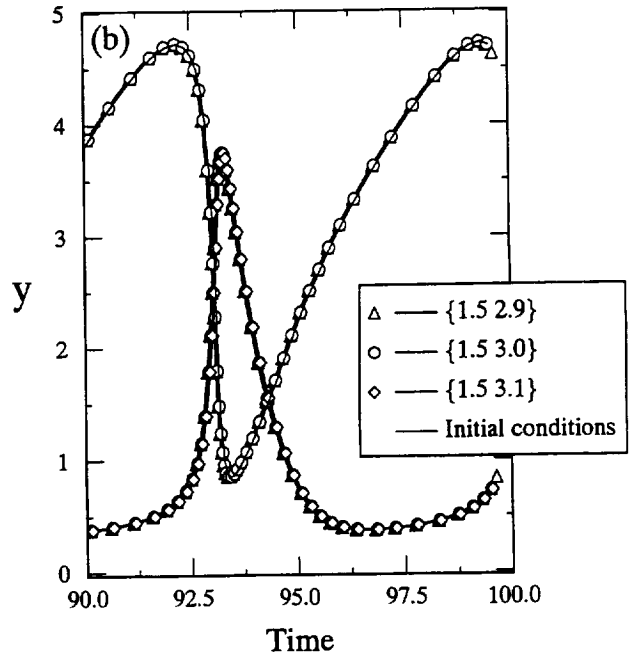


Figure 3.2: Chemical reaction: variations in chemical concentration (a) and residual error (b) obtained by semi-implicit integrator FGA in solution of Brusselator.

Figure 3.3: Chemical reaction: variation in chemical concentration of Brusselator for perturbed initial conditions obtained by FGA at short (a) and long (b) times.

ratio of the eigenvalues, a much stiffer set of equations ensues. For this test, A was kept as before and B was increased to 100, for which case the resulting eigenvalue ratio becomes $|\lambda_{max}|/|\lambda_{min}| = 9593$. The solutions, which are now asymptotic rather than periodic, are shown in figures3.4 and 3.5. As may be verified, for the same accuracy FGA proceeded with larger steps than SKA, especially within the stiff region. The total number of steps needed for FGA and SKA were 20 and 89, respectively. The explicit algorithm (i.e., EFA) required 14 823 steps to successfully complete the process! This result demonstrates clearly the advantage of using stiff integrators for stiff problems.

3.2 The Lorenz Chaos Problem

The following discussion deals with the now famous set of hydrodynamic equations developed by Lorenz (1963) for simulating, as simply as possible, the complex behavior of Earth's weather. Lorenz's set of ordinary differential equations has been analyzed in some detail by Hairer et al. (1993, pp. 120–125). We do not delve into the mathematical rigors of chaos theory; rather, we reproduce Lorenz's results by using the FGA algorithm and compare its numerical features with those of similar integrators.

The most significant feature of this problem is the *nonperiodic* behavior of its solution. It should be emphasized that the solution is continuous with continuous derivatives, unique, and bounded. Even so, although it is periodic, it never repeats its own history exactly. Its determinism is a transient property that disappears as time goes by. This set of differential equations serves as a contrast example to the Brusselator, and therefore it is a good second test case for our study.

Like the Brusselator, analyses were performed for both short and long times, with perturbed initial conditions, and for different global tolerances. This procedure was intended to distinguish between problem characteristics on the one hand and algorithmic behavior on the other. The set of mathematical equations given by Lorenz (1963) is as follows:

$$\left.\begin{aligned} \dot{X} &= -\sigma X + \sigma Y \\ \dot{Y} &= -XZ + rX - Y \\ \dot{Z} &= XY - bZ \end{aligned}\right\}, \qquad (3.2)$$

where σ, r, and b are related to the flow parameters, X is proportional to the intensity of convective motion, Y is proportional to the temperature difference between ascending and descending currents, and Z is

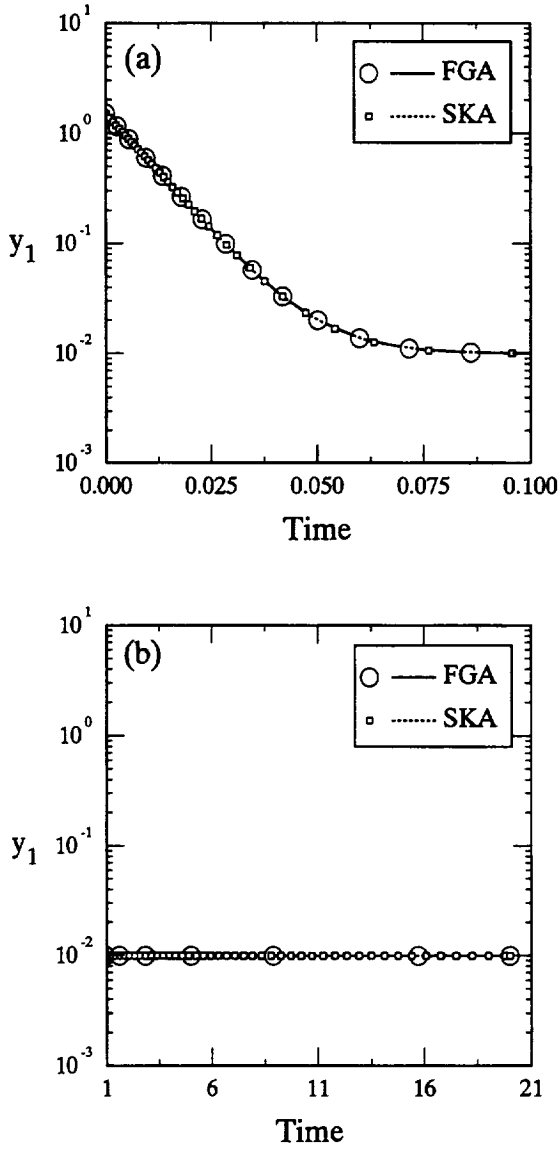


Figure 3.4: Chemical reaction: variation of y_1 under stiff conditions for short (a) and long (b) times where $A = 1$ and $B = 100$.

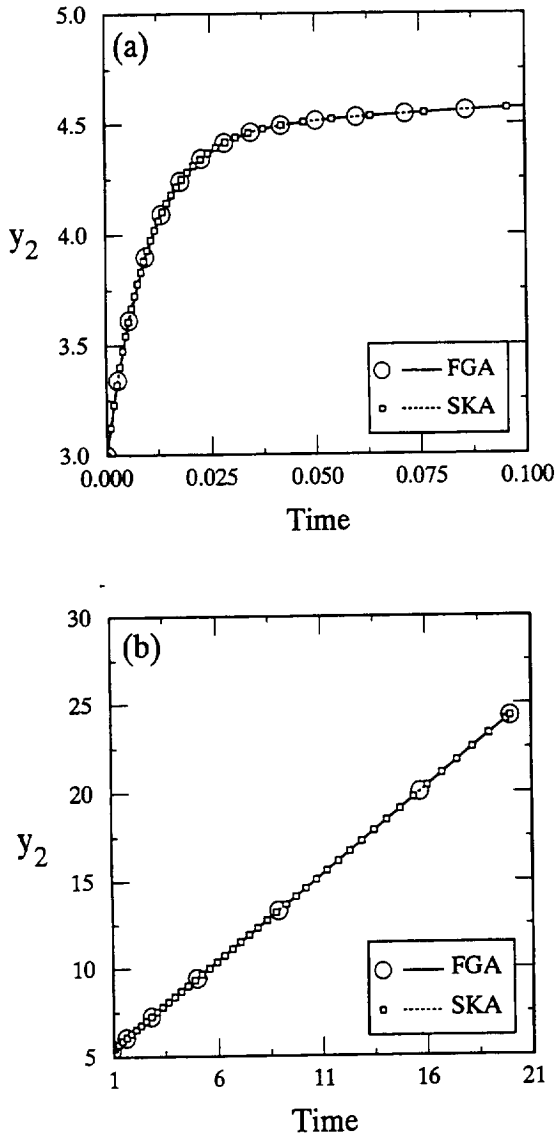


Figure 3.5: Chemical reaction: variation of y_2 under stiff conditions for short (a) and long (b) times where $A = 1$ and $B = 100$.

proportional to a distortion in the vertical temperature profile from linearity. Similar signs in X and Y imply that warm air is rising and cool air is falling. A positive value of Z indicates that the strongest temperature gradients occur near the boundaries—the base of a thunderhead, for example, and the Earth.

There are three steady states: $\{0 \quad 0 \quad 0\}^T$, which is unstable for values of $r > 1$, and $\{X = Y = \pm\sqrt{b(r-1)}, \quad Z = r-1\}^T$, which are unstable for values of $\sigma > (b-1)$ and of $r > r_c$ with $r_c = \sigma(\sigma+b+3)/(\sigma-b-1)$. These inequalities follow from the characteristic equation

$$\lambda^3 + (\sigma+b+1)\lambda^2 + (r+\sigma)b\lambda + 2\sigma b(r-1) = 0$$

which produces three *constant* eigenvalues. The flow parameters were chosen, as cited in the references, to be given by

1. $\sigma = 10$, $b = 8/3$, and $r = 28 > r_c$, in which case $r_c = 24.45$, leading to the steady-state points $\{-8.48 \quad -8.48 \quad 27\}^T$ and $\{8.48 \quad 8.48 \quad 27\}^T$
2. Initial conditions of $\{0 \quad 1 \quad 27\}^T$ and $\{-8 \quad 8 \quad 27\}^T$
3. A final (normalized) time of 5, extended to 100

For these values of σ, b, and r, one eigenvalue turns out to be real while the other two are complex, both having positive real components, and therefore the system is unstable. The ratio of their magnitudes is approximately 1.2:1, which does not indicate any stiff behavior.

The first analysis was performed with a shorter range of time. The three state planes are plotted in figure3.6. The FGA produced the same solution obtained by Lorenz, Hairer et al., and many others. The solution crossed the state planes several times as it circumnavigated the steady-state points, as expected. For the two different initial conditions, the solution followed two different trajectories with different ending points. From the numerical point of view, FGA required 208 steps, SKA required 13 905 steps, and EFA experienced unrestrained error growth.

The magnitude of the residual errors for FGA and SKA are plotted in figure3.7. In this case, SKA ended up with better accuracy than was initially required. The reason is the much smaller step size mandated by the algorithm. On the other hand, for FGA the step size was about right to meet the external tolerance.

The next set of numerical tests dealt with small perturbations in one coordinate of the initial value. In the application of systems of differential equations, such perturbations may represent inaccuracies in laboratory measurements, which are inevitable phenomena. To this end, the analyses were performed for

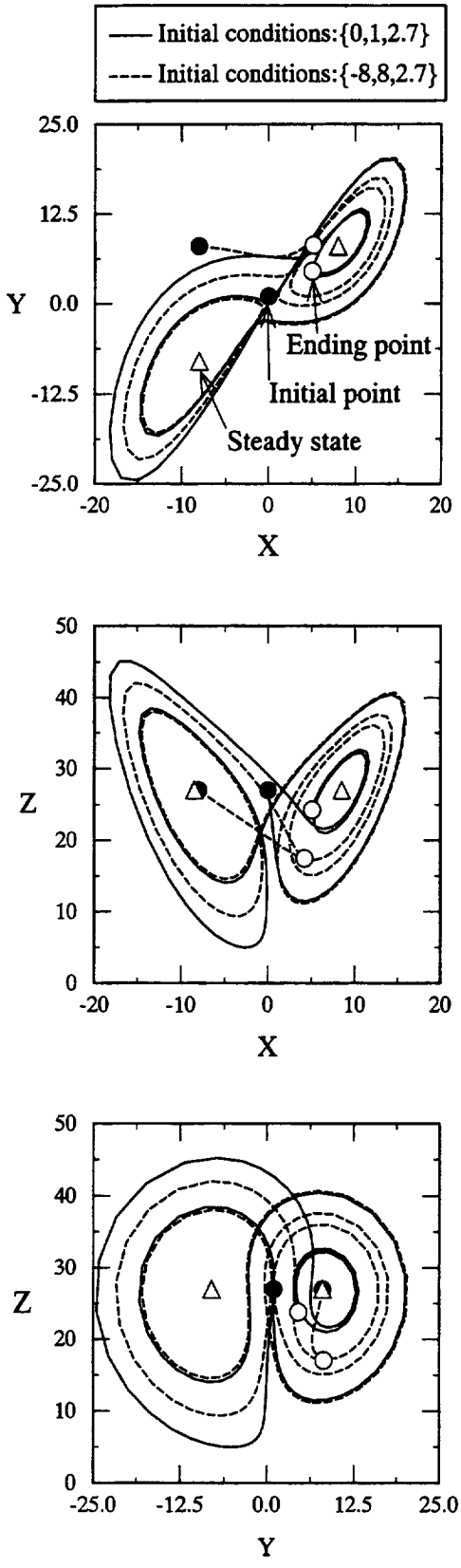


Figure 3.6: Lorenz chaos problem: X–Y, X–Z, and Y–Z spaces, solved for two different initial conditions. Tolerance, 1×10^{-4}; final time, 5.

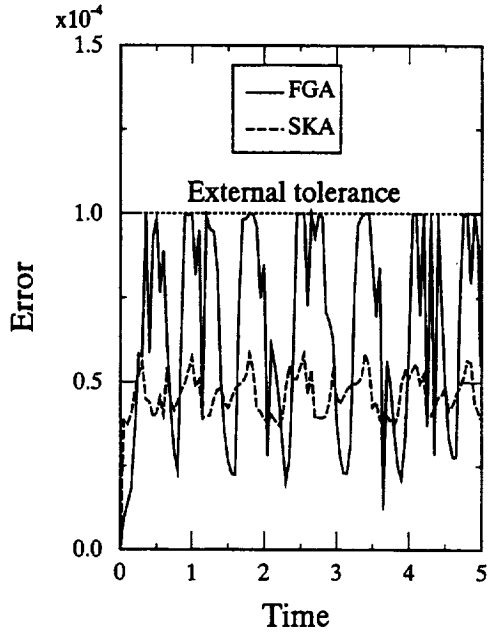


Figure 3.7: Lorenz chaos problem: residual errors obtained for two Rosenbrock algorithms.

three initial conditions: $\{1 \quad 0 \quad 3.9\}^T$, $\{1 \quad 0 \quad 4.0\}^T$, and $\{1 \quad 0 \quad 4.1\}^T$. In addition, the final time was increased to 100. As can be seen in figures3.8 and 3.9, the final state was totally different for each of the three cases, even though the initial point was perturbed in one component by only 2.5%. This agrees with Lorenz's conclusion: for a chaotic system, any deterministic prediction is accurate only in the near future. Even the slightest differences in the starting point result in vastly different states when predicted far into the future. This is the reason why the weather forecast on your evening news typically goes out to only five days, with a decrease in reliability the further out one goes in the forecast.

The last test run on this problem involved changing the error tolerance and observing the behavior of the algorithm itself. Being a nonstiff type of problem, the appropriate step size is accuracy driven. In other words, the question asked here is, How sensitive are chaotic phenomena to the error tolerance of our integrator? All numerical integrators have a sensitivity in this respect; it is in the extent of the effect that they differ. Said differently, truncation error is one source leading to a loss in determinism in a chaotic system of equations; perturbed initial conditions are another source. Any numerical analysis in chaos should distinguish between these two sources if accurate near-future predictions are to be made.

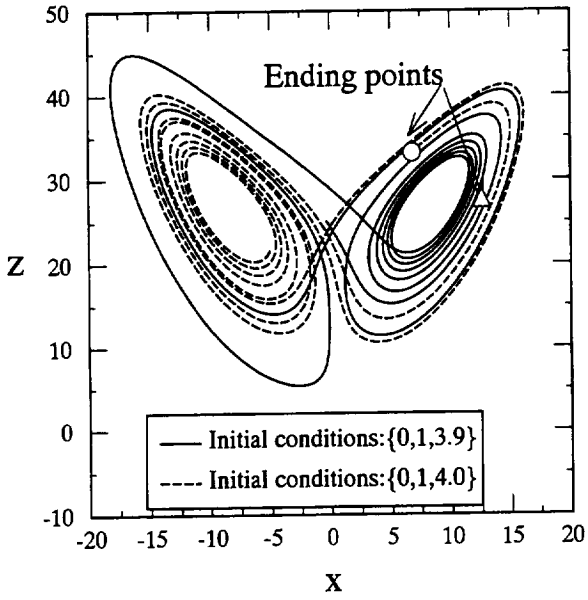


Figure 3.8: Lorenz chaos problem: solutions in X–Z space for perturbed initial conditions. Tolerance, 1×10^{-4}; final time, 100.

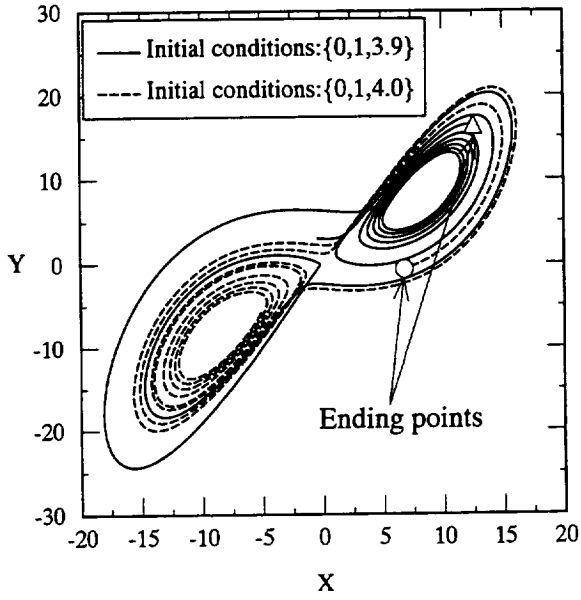
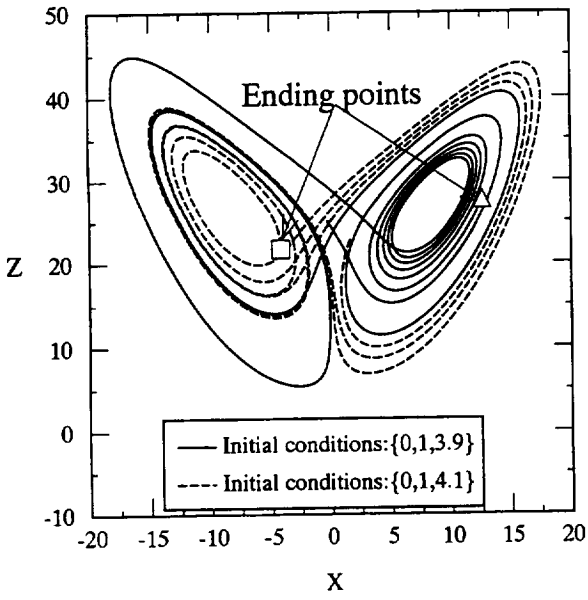


Figure 3.9: Lorenz chaos problem: solutions in X–Y space for perturbed initial conditions. Tolerance, 1×10^{-4}; final time, 100.



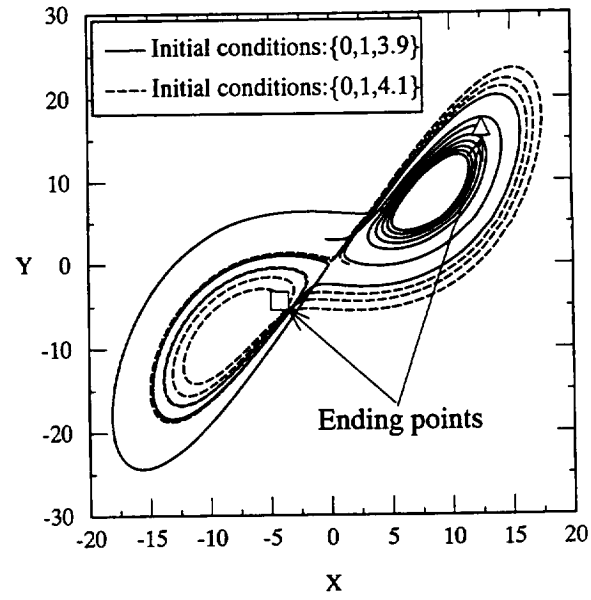


Figure 3.10 shows the control of the externally applied tolerances over the performance of the FGA numerical scheme. The residual errors did not exceed the imposed limits. The upper error bound depended on the externally enforced values for both short and long time intervals.

Because different tolerances produce different time steps, it is difficult to say with certainty how similar or dissimilar the numerical solutions really are. Nevertheless, figure3.11 shows these differences to be quite small, at least in the near future. However, the variance is not always this small. As time grows, so does the difference between solutions calculated with different error tolerances. Even so, this integrator seems to be less sensitive to error tolerance than other integrators we have used. By the time of 100, the errors due to variations in the truncation error have not yet produced solutions that have significantly branched away from one another. This is not the case for small variations in the initial conditions.

3.3 A Singular System

In the following example, the integrators were applied to solve an *a priori* singular system. These equations were taken from Enright and Pryce (1987, case D4) (see also Press et al., 1992, p. 734) and are given by

$$\left.\begin{aligned} \dot{y}_1 &= -0.013y_1 - 1000y_1y_3 \\ \dot{y}_2 &= -2500y_2y_3 \\ \dot{y}_3 &= -0.013y_1 - 1000y_1y_3 - 2500y_2y_3 \end{aligned}\right\}. \quad (3.3)$$

As can be easily seen, the last equation is the sum of the first two, which means that the system is singular and therefore has an eigenvalue of zero. For this system, the steady state is calculated to be $y_1 = y_2 = 0$ $\forall\ y_3$, with the remaining two eigenvalues being positive whenever $y_3 < 0$, leading to instability.

The analysis parameters were chosen exactly as in Press et al.; namely, the initial conditions were set to $y_1 = 1$, $y_2 = 1$, and $y_3 = 0$, and the final time was set to 50. The analyses were performed for two global tolerances (viz., 10^{-4} and 10^{-3}) by the two semi-implicit Runge-Kutta integrators (i.e., FGA and SKA). The solutions acquired for the three components, y_1, y_2, and y_3, are plotted in figure3.12. Both algorithms provided solutions within the required global tolerance. The major differences lie in the numerical features of each method.

When using the tighter global tolerance (i.e., 10^{-4}), FGA ran to completion using 19 steps; the SKA algorithm needed 86 steps. In contrast, the explicit integrator EFA oscillated about the initial condition until the analysis was artificially stopped. The

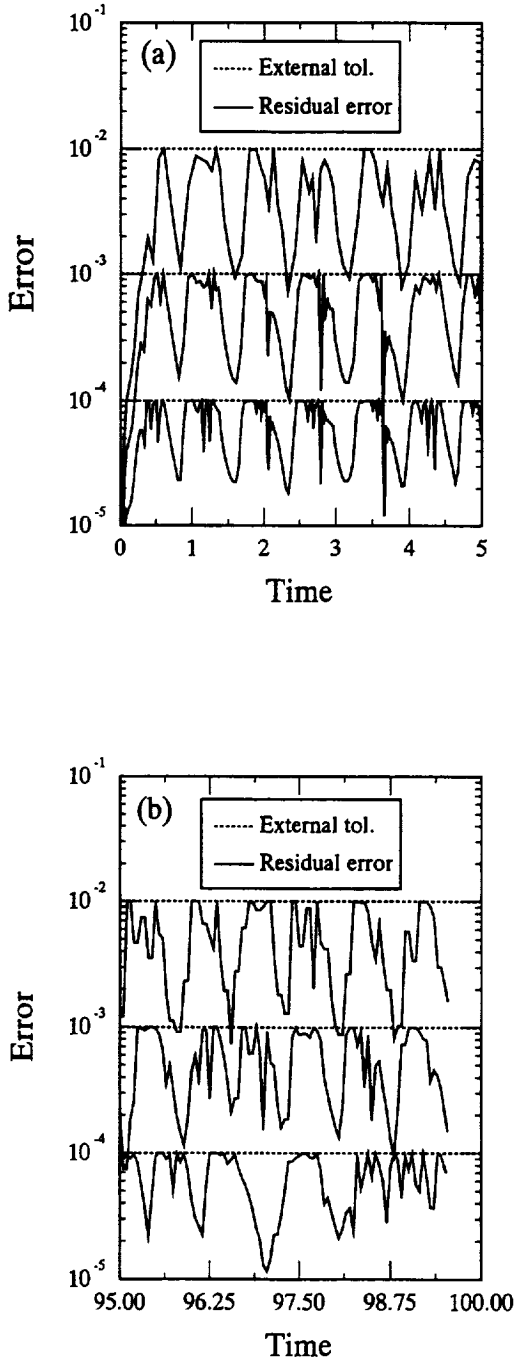
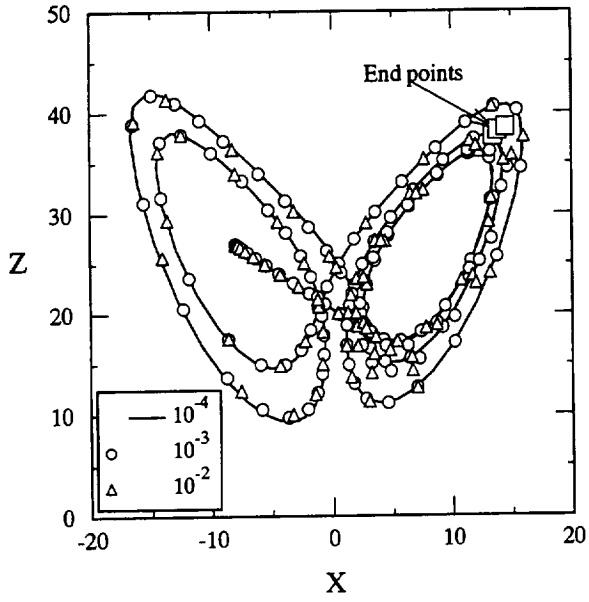


Figure 3.10: Lorenz chaos problem: residual errors of FGA for various global tolerances over short (a) and long (b) time intervals.



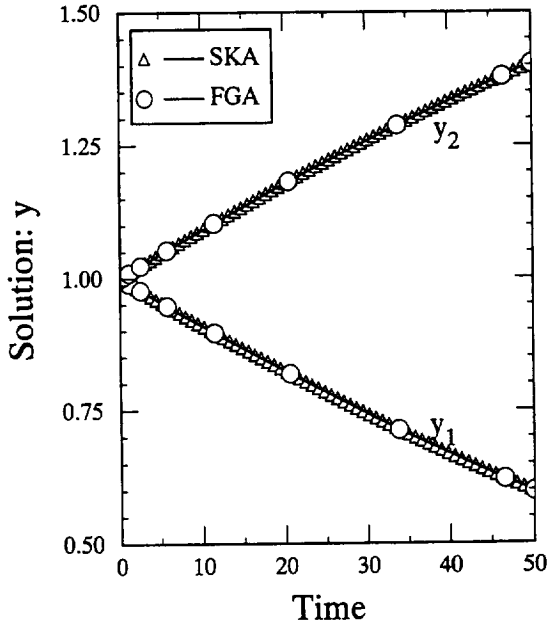
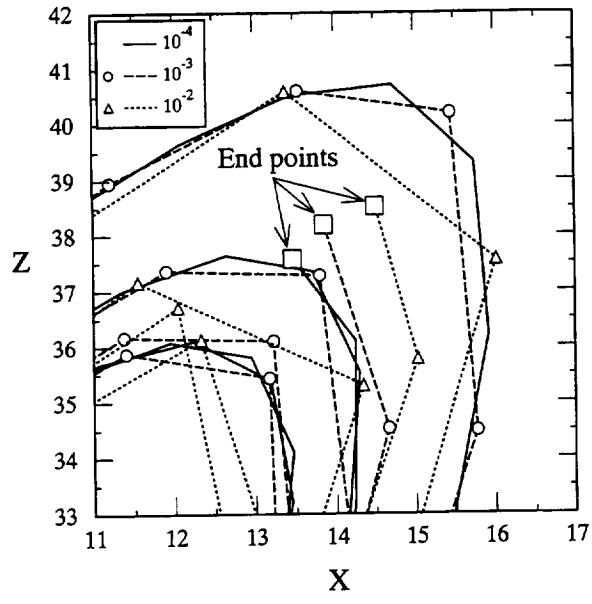


Figure 3.11: Lorenz chaos problem: solutions in X–Z space for various global tolerances. Initial condition, $\{-8\ 8\ 27\}^{\mathrm{T}}$; final time, 5.



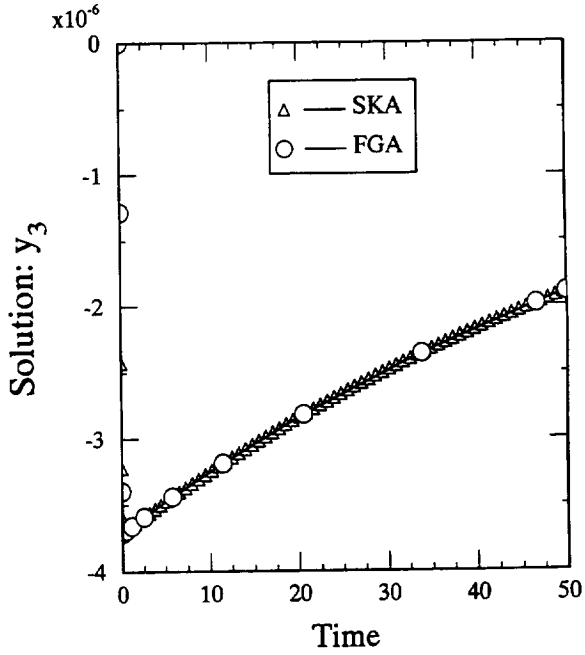


Figure 3.12: Singular system: solutions obtained by semi-implicit Runge-Kutta methods.

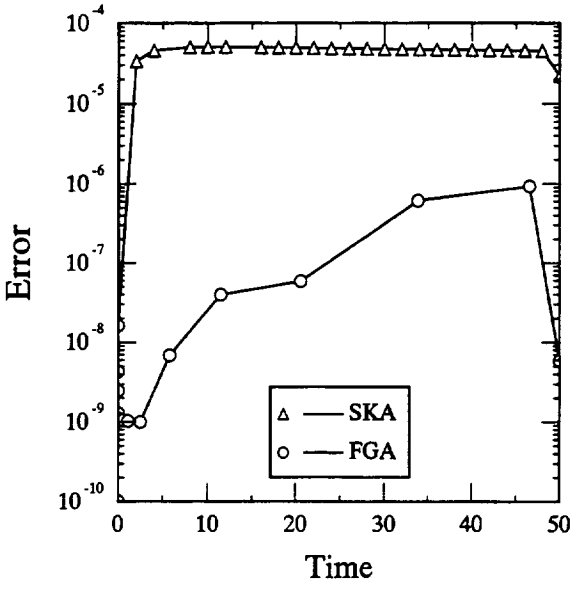


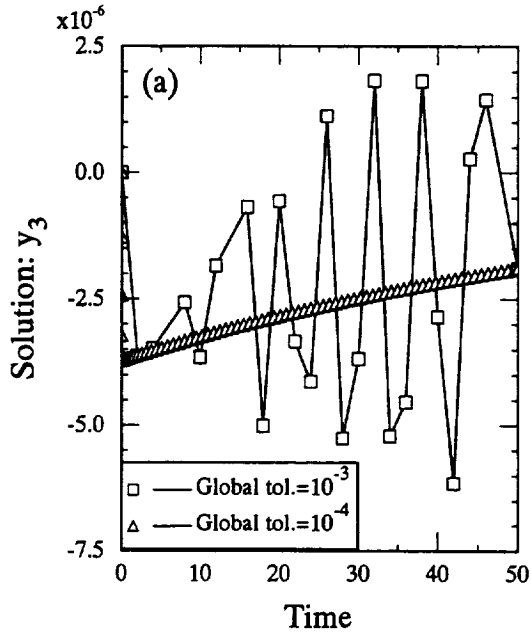
Figure 3.13: Singular system: truncation errors of two Rosenbrock methods.

residual errors for the two semi-implicit algorithms are shown in figure3.13. Despite the reduced step size in SKA, its error was greater by a few orders of magnitude than that of the FGA integrator.

Upon imposing the second global tolerance (i.e., 10^{-3}), the number of steps required by SKA was reduced to 34, but the accuracy of the solution deteriorated as well. The results for this analysis are plotted in figure3.14. When applying the FGA using this tolerance, the number of steps was reduced by 2, to 17, with no apparent differences between the solutions.

The final consideration given to this example was monitoring the Jacobian norm, plus investigating its influence upon the performance of the algorithms. Up to this point, all the analyses were performed *without* restraining the norm $\|\mathbf{J}\|$ at all. In this case, with a tolerance of 10^{-4} imposed, FGA completed its analysis in 19 steps, with the Jacobian norm ranging between 0.846275 and 795 500, as shown in figure3.15. SKA required 80 steps with its norm, in this case, varying from of 0.846275 to 6691. Both algorithms provided accurate results despite the extreme values of their coefficient matrices.

In contrast, when the norm of the Jacobian was limited to have a maximum value of 10, the total number of steps for FGA went up to almost 40 000. In all the cases, as long as the matrix $[\mathbf{I}-bh\mathbf{J}]$ was not singular, these semi-implicit algorithms converged,



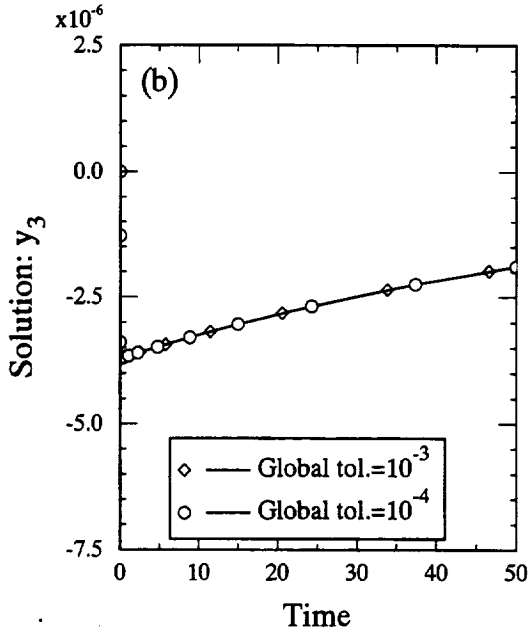


Figure 3.14: Singular system: y_3 component for two global tolerances. (a) SKA. (b) FGA.

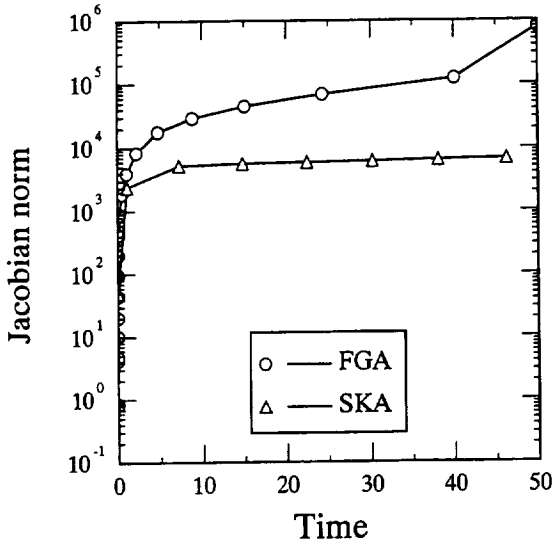


Figure 3.15: Singular system: variations in norms of Jacobian for FGA and SKA methods.

and as fast as their norms allowed them to. It therefore seems best not to interfere with the Jacobian at all.

3.4 Viscoplasticity

Viscoplastic constitutive equations describe the rate-dependent response of metals at high temperatures. They usually incorporate two competing mechanisms (viz., hardening and recovery processes) that are in balance with each other upon reaching a steady state. The models were mathematically defined by using the same type of equations as employed in the previous examples. One characteristic of viscoplasticity is that the eigenvalues are nonnegative and strongly state dependent (Arnold, 1990). The result is a stable system with regional stiffness.

The material model used in the current study was developed by Robinson. Details of the formulation may be found in Robinson (1978) and Robinson and Swindeman (1982) and are summarized in appendix B. In addition to the typical regional stiffness of viscoplastic models, this model includes a step function to simulate abrupt changes in the internal state behavior, which are observed to occur in the dislocation network at load reversals. This step function makes the model especially attractive from the numerical point of view, in the sense that it is a challenging test case for study.

In essence, a viscoplastic model contains three types of equation: (1) Hooke's law defining the relationship between stress (load) and strain (displacement) (i.e., the input-output pair); (2) a flow equation defining the kinetics of inelastic deformation—the primary source of nonlinearity; and (3) an evolution equation quantifying a change in internal state, in this case, a back stress resulting from the heterogeneous distribution of dislocations.

For the current implementation, these equations have been rearranged such that

1. First, a deviatoric system is solved for two independent state variables, specifically, the effective stresses and back stresses denoted as $\mathbf{\Sigma}, \mathbf{A} \in \Re^{3\times 3}$, which are symmetric (i.e., $[*] = [*]^{\mathsf{T}}$) and deviatoric (i.e., $\mathbf{I} : [*] = 0$).

2. Then, the reversible, isotropic, hydrostatic component H_{ydro} is added in the calculation of Cauchy's (applied) stress: $\boldsymbol{\sigma} = \mathbf{\Sigma} + \mathbf{A} + H_{ydro}\mathbf{I}$.

The final system of equations has the matrix form

$$\begin{Bmatrix} \dot{\mathbf{\Sigma}} \\ - \\ \dot{\mathbf{A}} \end{Bmatrix} = \begin{bmatrix} \mathbf{M}_{11} & | & \mathbf{M}_{12} \\ - - - & - & - - - \\ \mathbf{M}_{21} & | & \mathbf{M}_{22} \end{bmatrix} \begin{Bmatrix} \mathbf{\Sigma} \\ - \\ \mathbf{A} \end{Bmatrix} + \begin{bmatrix} \mathbf{D}^{el} \cdot \left[\mathbf{I} - \mathbf{I}(\frac{1}{3}\mathbf{I}:)\right] \\ - - - - - - - \\ \mathbf{O} \end{bmatrix} \begin{Bmatrix} \dot{\boldsymbol{\epsilon}} \\ - \\ \mathbf{O} \end{Bmatrix}, \quad (3.4)$$

where $[\mathbf{M}_{ij}] \in \Re^{3\times 3}$. The exact formulation is detailed in appendix B.

For the following analyses, the total strain $\boldsymbol{\epsilon}$ was taken to be the externally enforced function and was considered to be cyclic (cf. fig. 3.16). The error tolerance was set to the usual value of 10^{-4}. The material constants were taken directly from Robinson's references. Both explicit and implicit solvers were used to obtain predicted material behaviors. All three integrators provided the correct solution, within the preassigned tolerance. To achieve this, FGA needed 185 steps, SKA needed 5573 steps, and EFA needed 79 726 steps. Figure 3.17 shows the variation of step size throughout the whole cycle, as obtained from the FGA integrator. The areas where point density is high are indicative of stiff zones, regions of strong nonlinearity, or locations dominated by the step functions.

The regions marked A and B in figure3.17 are the locations where the step function is active. Because of the short duration of its application and the abrupt change that it causes, the step size is significantly reduced by all algorithms when they encounter this

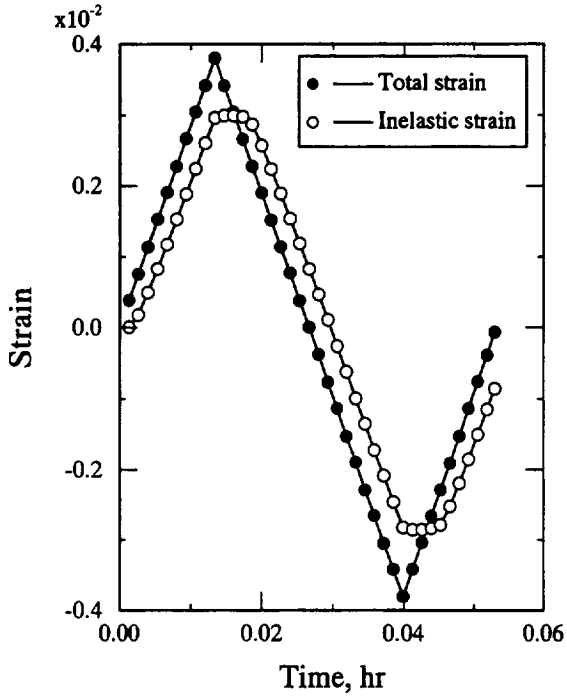
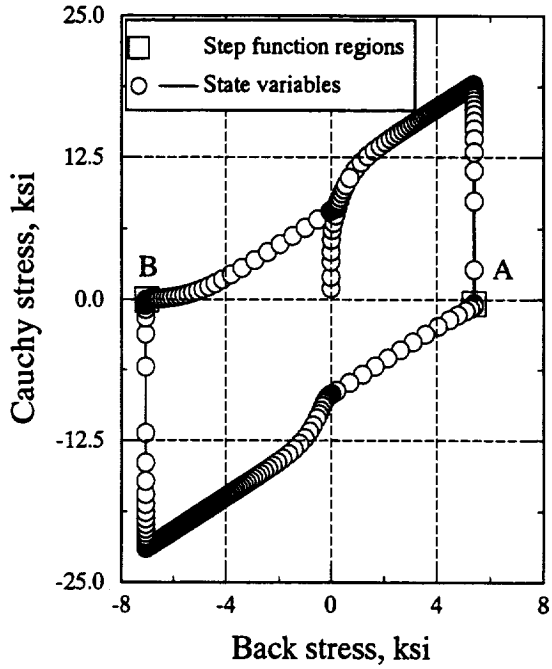


Figure 3.16: Robinson's viscoplastic material: input (total) and output (inelastic) strains.



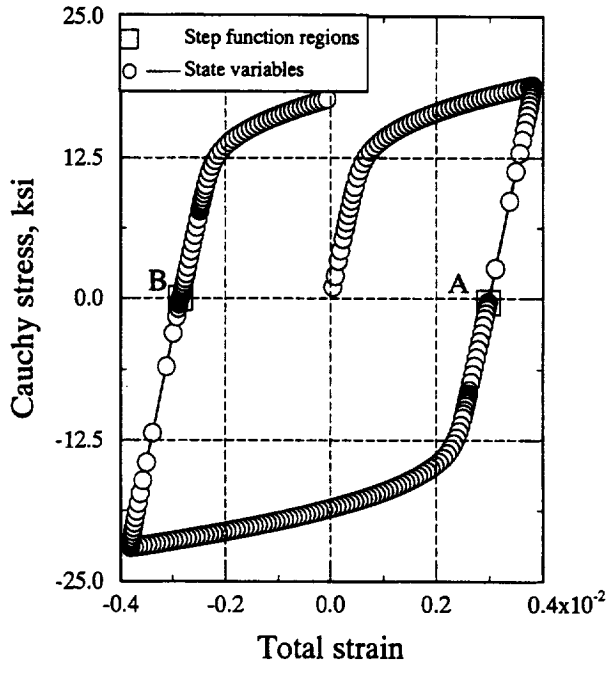


Figure 3.17: Robinson's viscoplastic material: various state spaces for variables.

step function, which is numerically simulated by a spline function. Region B of figure3.17 is blown up in figure3.18 to illustrate this phenomenon, as obtained by FGA. The other two algorithms had to proceed with much smaller steps through this region to end up with the same solution accuracy.

Figure 3.19(a) shows the step-size variation through the knee of the curve in the first quadrant of state space for both semi-implicit algorithms. Here the solution is accuracy driven rather than stability driven; in other words, here the solution is not stiff. *Our integrator, unlike most stiff integrators, is efficient for nonstiff problems, too.* In contrast, figure 3.19(b) shows step-size variation in the region where stability drives the solution and accuracy has little significance. After reaching a Cauchy stress of nearly 20 ksi, the load was removed and consequently the Cauchy (applied) stress fell off. The step sizes were small during unloading, especially for SKA. The problem was at maximum stiffness in the region of maximum Cauchy stress. Stiffness did not diminish much until the applied stress dropped below the level of the internal stress. As concluded by Arnold (1990), this region is susceptible to numerical stiffness and therefore is a worthwhile location for comparing algorithms.

The last feature that we investigated in this example was the effect due to variation in the Jacobian.

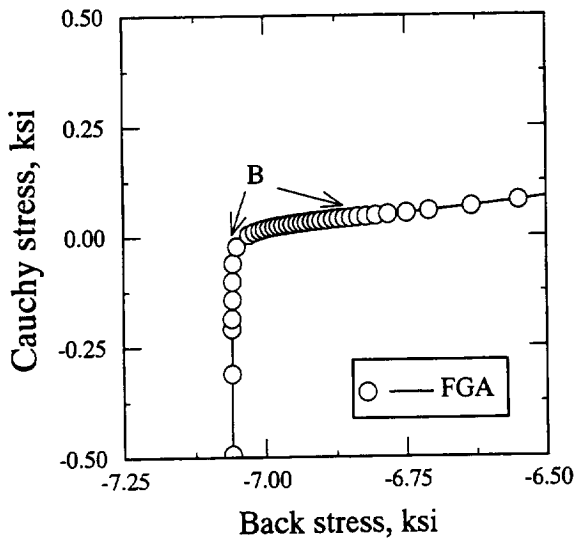
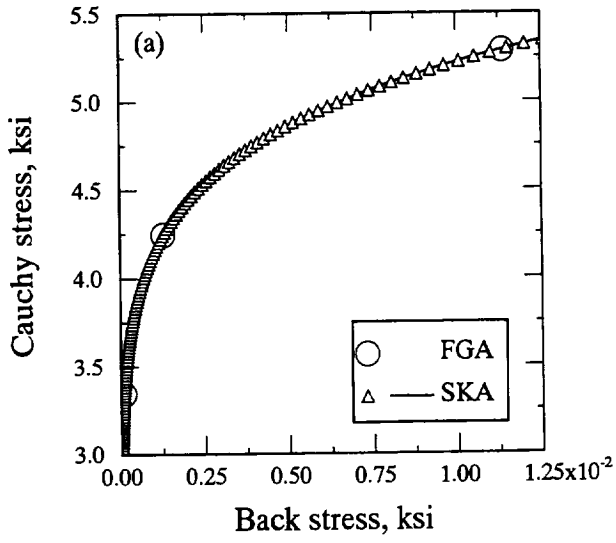


Figure 3.18: Robinson's viscoplastic material: State response through region (B) where step function is applied.

As mentioned by Shampine (1980), Lambert (1991, pp. 251–254), and many others, the Jacobian dominates implicit algorithms and is a crucial factor in obtaining the correct solution. Therefore, the Jacobian norm (viz., $\|\mathbf{J}\|_1 \stackrel{\text{def}}{=} \max_i(\sum_j |J_{ij}|)$) was monitored in both the FGA and SKA algorithms. Their outputs are plotted in figure3.20.

Figure 3.20(a) shows variations in the Jacobian norm obtained from FGA for different prescribed maxima of the Jacobian. The absolute value of the Cauchy stress is plotted in the background as an aid in visually identifying the stiff and nonstiff regions. Analyzing this plot, it can be seen that the extremes occurred near the steady states. The flatter the stress (output)–versus–time curve, the greater the value of the Jacobian. The maximum occurs in the third quadrant of state space, with $\|\mathbf{J}\| = 16.7$. The lesser peaks in these Jacobian time plots are identified with the step function becoming active.

When an upper bound was imposed on the Jacobian norm beyond which the step size must be reduced, the previously mentioned peaks became chopped off and the norm was maintained constant until the solution left that particular stiff region. Naturally, bounding the norm affected the number of integration steps. The lower the bound, the greater the number of steps. Without any user interference (un-



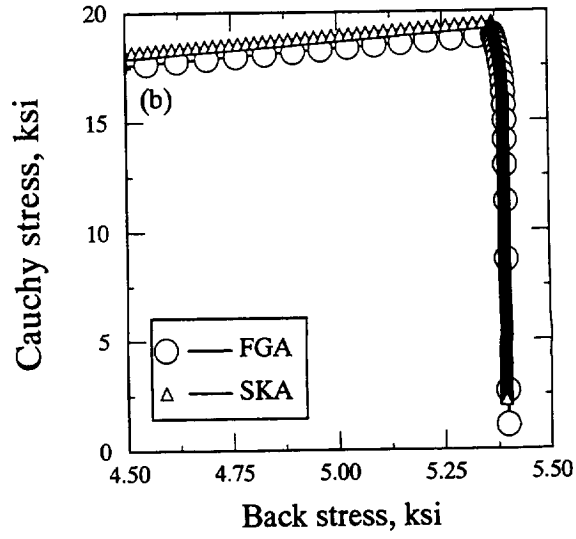
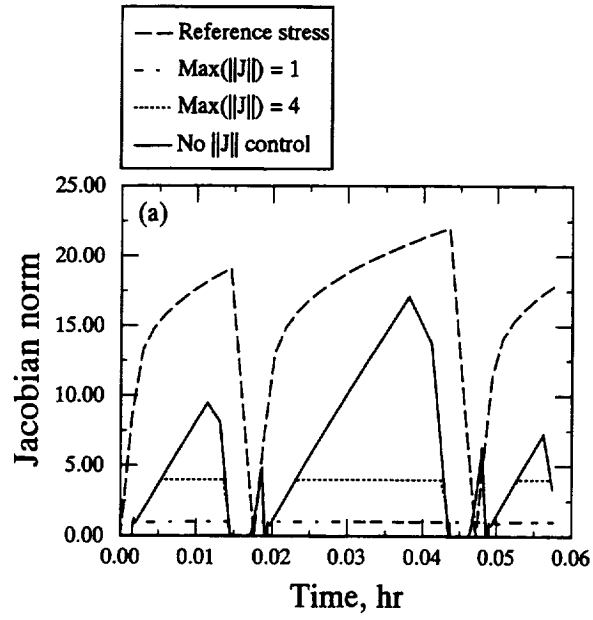


Figure 3.19: Robinson's viscoplastic material: variations of step size for FGA and SKA in (a) transient nonstiff region and (b) stiff region near steady state.



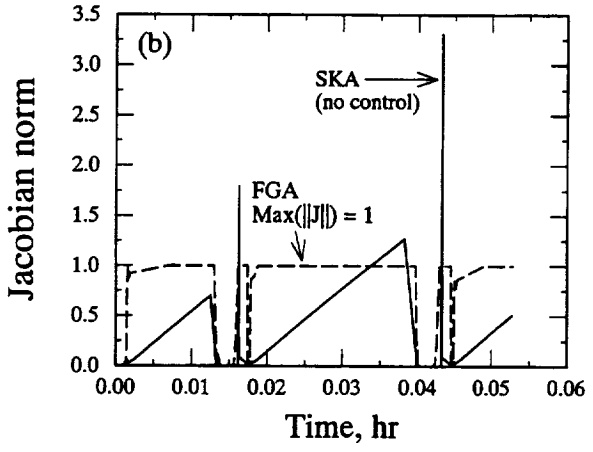


Figure 3.20: Robinson's viscoplastic material: variation of $\|\mathbf{J}\|_1$ for (a) FGA and (b) both FGA and SKA.

bounded), the number of steps for the whole cycle was 185 for FGA. When the norm of the Jacobian was constrained to be no greater than 5, 3, 2, and 1, the number of steps for the FGA integrator increased to 211, 248, 303, and 487, respectively.

Figure 3.20(b) shows a comparison between SKA and FGA. For SKA, with unbounded magnitudes of the norm, the Jacobian norm was almost always smaller than the FGA norm, even at its lowest clipped value (investigated) of 1. The maximum value over the entire cycle was $\|\mathbf{J}\| = 3.5$ for the SKA algorithm, and it happened during an active period of the step function. Peaks are apparent in the stiff regions as well, but their values are smaller relative to those of FGA, especially when it was not clipped.

The solutions acquired in the stiff regions, obtained both with and without constraining the Jacobian, are plotted in figures3.21 to 3.23 for the various phase planes. Limiting the Jacobian is seen to have had almost no effect on the accuracy of the results, which agrees with a conclusion obtained from the prior example. The maximum relative error observed between the solution without bounding the Jacobian and the one obtained with $\max(\|\mathbf{J}\|) = 1$ was 1%.

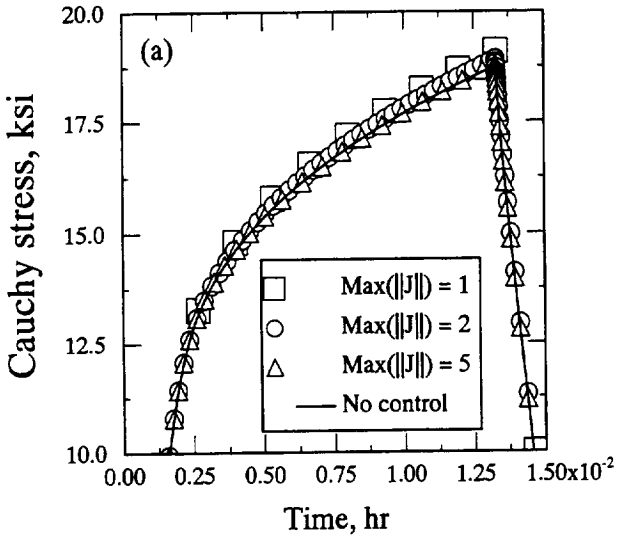
3.5 Stiff Chemical Kinetics

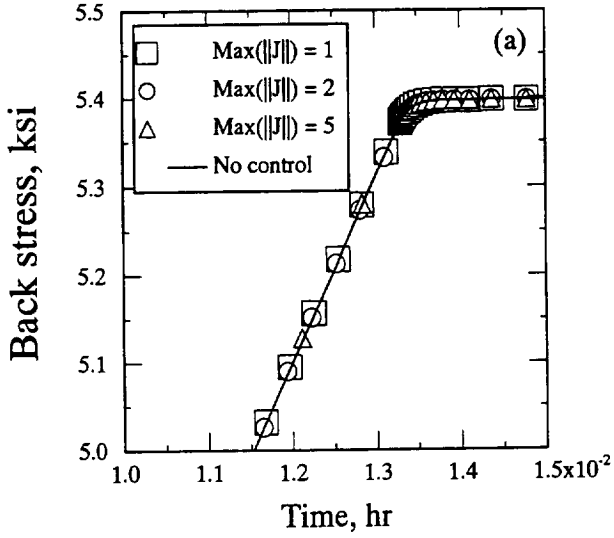
In our final example, we return to the field of chemical kinetics, as the related problems are known to be among the most difficult to deal with from the numerical point of view. The following example is test case F1 in Enright and Pryce (1987). No details are provided concerning the physical interpretation; nevertheless, the given system exhibits extremely stiff behavior. Therefore, it was intriguing for us to examine the performance of the two semi-implicit algorithms.

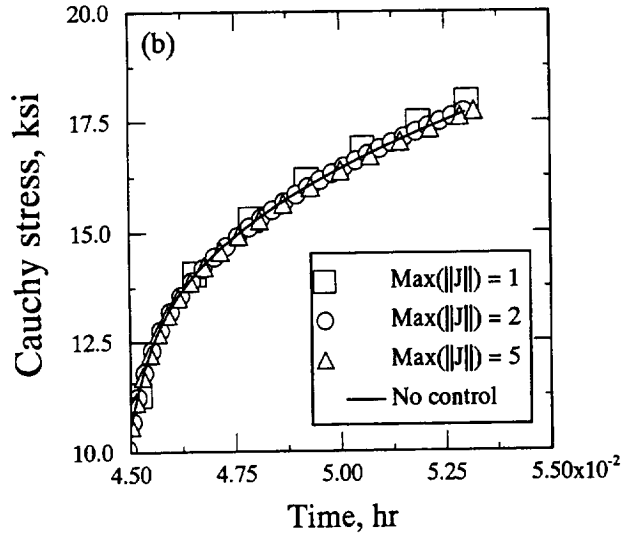
To more accurately single out the differences between these two integrators, time-step control of the SKA algorithm was switched over to that of Gustafsson, thereby putting them on a more equal footing for comparison purposes. That is, Shampine's parameters (Shampine, 1982) were combined with Gustafsson's error routine (Gustafsson et al., 1988) to create a solver, which is abbreviated as SGA.

The pertinent equations for this chemical kinetics problem are

$$\left.\begin{aligned} \dot{y}_1 &= 1.3(y_3 - y_1) + 10\,400\ k\ y_2 \\ \dot{y}_2 &= 1880[y_4 - y_2(1+k)] \\ \dot{y}_3 &= 1752 - 269y_3 + 267y_1 \\ \dot{y}_4 &= 0.1 + 320y_2 - 321y_4 \end{aligned}\right\} \tag{3.5}$$







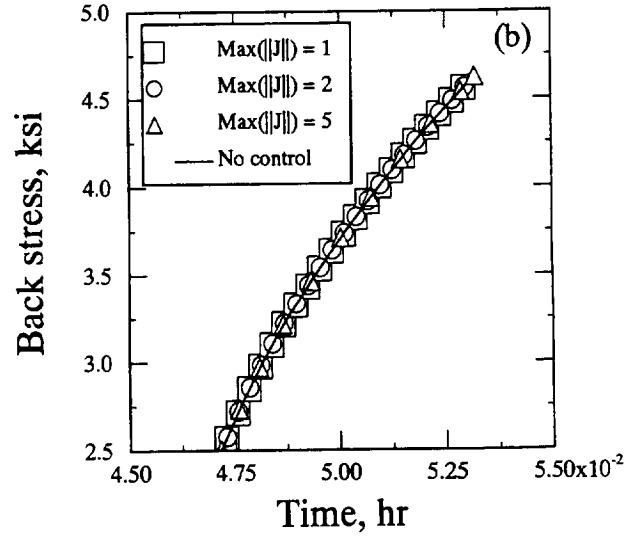


Figure 3.21: Robinson's viscoplastic material: external stress versus time at first reversal (a) and at end (b) of loading history.

Figure 3.22: Robinson's viscoplastic material: internal stress versus time at first reversal (a) and at end (b) of loading history.

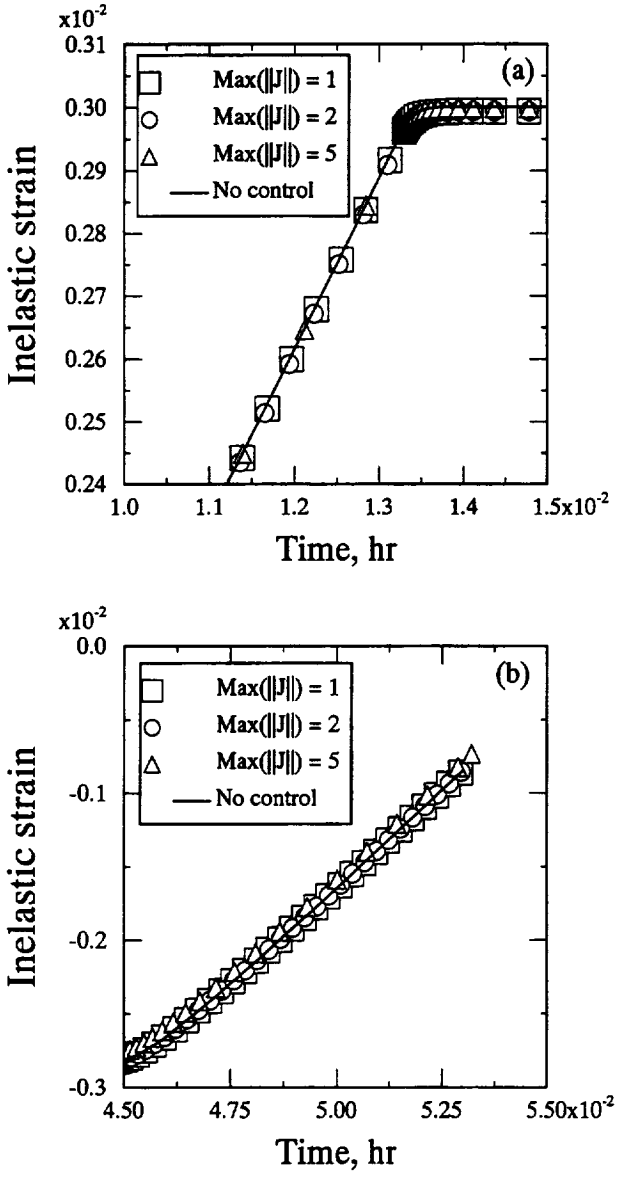


Figure 3.23: Robinson's viscoplastic material: plastic strain versus time at first reversal (a) and at end (b) of loading history.

with the parameter

$$k = \exp(20.7 - 1500/y_1).$$

The final analysis time was set to 1000, with the initial condition set to $\mathbf{y} = \{761 \quad 0 \quad 600 \quad 0.1\}^T$. The Jacobian is described by the matrix

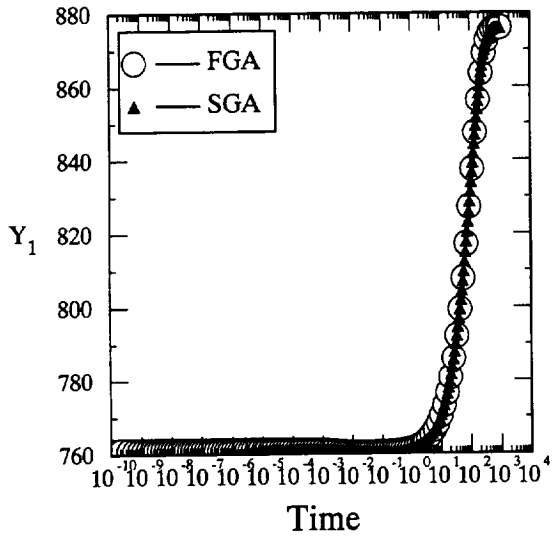
$$\begin{bmatrix} -1.3 & 10\,400 & 1.3 & 0 \\ -1880y_2\left(\frac{\partial k}{\partial y_1}\right) & -1880(1+k) & 0 & 1880 \\ 267 & 0 & -269 & 0 \\ 0 & 320 & 0 & -321 \end{bmatrix}$$

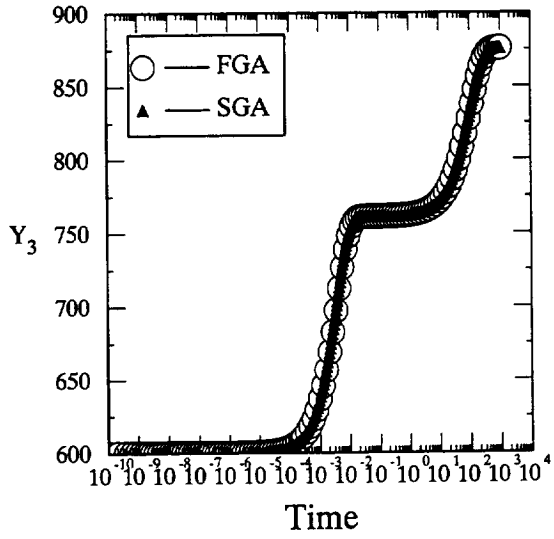
Solving the resulting fourth-order characteristic equation $|\mathbf{J} - \lambda\mathbf{I}| = 0$ leads to the four negative eigenvalues $\{-2.56 \times 10^{11} \quad -321 \quad -271 \quad -.0096\}$, with a resulting eigenvalue ratio on the order of $|\lambda_{max}|/|\lambda_{min}| \approx 10^{13}$. Obviously, this system is stable and extremely stiff.

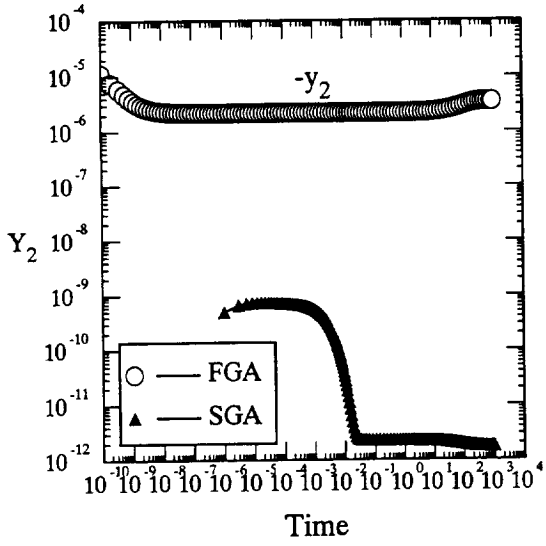
The four components of the solution are plotted against time in figures3.24 and 3.25. The results provided by both algorithms are accurate, as long as components y_1, y_3, and y_4 are considered. The converged solutions of both solvers follow the same patterns. Unlike all previous examples, the total numbers of integration steps for both formulations were about the same; specifically, SGA converged after 170 steps and FGA after 117. There is, however, a large discrepancy in the initial time-step size used by the two algorithms: SGA began with a step size of about 10^{-6}; FGA began at about 10^{-10}.

Consistency between predictions obtained by FGA and SKA was different for the variable y_2. Shampine's formulation provided what we believe to be the more reasonable solution. His algorithm predicts a change at the beginning of the analysis and then an asymptotic approach to zero. The prediction from our algorithm, albeit near zero, was negative—a fact that makes no physical sense if one considers the y_i to be measures of chemical concentration. Again, because we do not know the physics behind this system of equations, this is speculation on our part.

Apparently, the FGA results accumulated an initial overshoot error that did not clear itself throughout the whole analysis. The resulting y_2 obtained small but negative values, which triggered our suspicion and led to the conclusion that, for this *extreme* stiffness, Shampine's parameters seem to work better—probably because his integrator is A-stable (stiffly stable) while ours is not. Also, both integrators are stable at infinity—his being damped while ours is not. In other words, Shampine's integrator has better stability properties than does ours, and this shows up in the presence of extreme stiffness.







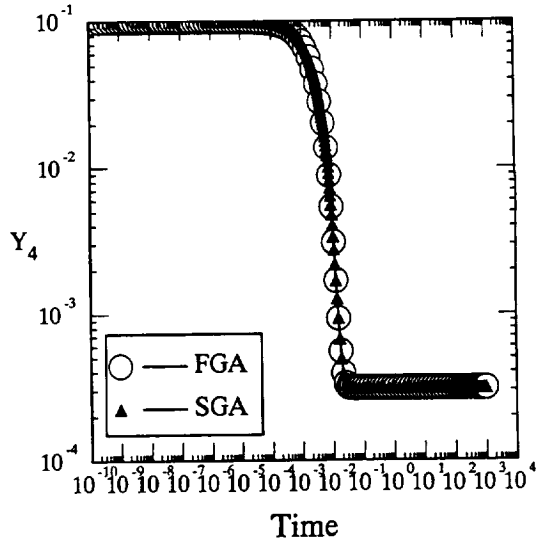


Figure 3.24: Chemical kinetics: semi-implicit algorithms with Gustafsson's error analysis for y_1 and y_2.

Figure 3.25: Chemical kinetics: semi-implicit algorithms with Gustafsson's error analysis for y_3 and y_4.

An important constraint in the overall solution process is that number of significant figures at which an analysis is done. Roughly speaking, the eigenvalue ratio for the stiffness of any nonsingular problem should not approach or exceed the number of significant figures used in an implementation. For this problem, the eigenvalue ratio was $\approx 10^{13}$ and FORTRAN double-precision variables were used. We believe that if higher-order precision would have been used, both algorithms would have produced the same answer. In other words, we believe that it is the numerical precision used, not the integrator itself, which determines that upper bound in stiffness ratio that can be safely handled by a stable Rosenbrock-Wolbrandt method.

Chapter 4

Summary

A Rosenbrock (1963) integrator of the Wolfbrandt (1978) type has been derived. This development took "the" classic Runge-Kutta integrator of Kutta (1901) and extended it to a Rosenbrock-Wolfbrandt integrator for the numerical integration of stiff systems. A fifth stage was added to produce a third-order embedded solution for estimating the error. The result is a semi-implicit 4(3)-order integrator appropriate for systems of ODE's that are stiff, unstable, or singular. It even performs competitively for well-behaved systems where explicit Runge-Kutta integrators have been the usual method of choice.

The capabilities of our integrator are probably best summarized in tabular form (viz., table 4.1). This table presents the number of steps required to successfully integrate to specified error tolerances for each of the five systems of ODE's investigated. Compared are the explicit 4(5) Fehlberg (1969) integrator (EFA), which is used in many commercial codes; the Rosenbrock integrator (SKA) of Shampine (1982), which is advocated by Press et al. (1992) for integrating stiff systems that are not excessively large in number (say, $M < 10$) and whose error tolerance is not too stringent ($\epsilon > 10^{-6}$); and the Rosenbrock integrator (FGA) derived herein.

Shampine's integrator, as reported in Press et al., uses the time-step controller of Kaps and Rentrop (1979); our integrator uses the time-step controller of Gustafsson et al. (1988). To eliminate discrepancies caused by the time-stepping algorithm, Shampine's integrator was linked with both controllers. The number of steps required for successful integration using the time stepper of Kaps and Rentrop has the column heading K.Err; results obtained using the time stepper of Gustafsson et al. have the heading G.Err. Looking at table 4.1, it is readily apparent that the Gustafsson et al. time stepper is superior to that of Kaps and Rentrop.

A fairer comparison now exists between SKA (using G.Err) and FGA. In all cases, except for the chemical kinetics problem of extreme stiffness, the FGA algorithm was superior to all others, including the Fehlberg integrator in the case of the nonstiff Brusselator.

Shampine (1985) made the following statement: "At present, codes are clearly intended for stiff or nonstiff problems, but not both. Deciding the type of the problem is an impossible task for a user. This author considers the question of how to relieve the user of this decision to be the most pressing question in the area of ODE mathematical software." The FGA integration algorithm is a possible solution to this problem. Unlike other nonstiff integrators, this algorithm seems to be efficient for stiff and nonstiff problems alike. Its good nonstiff properties likely come from the fact that FGA extends a proven integrator of Kutta (1901), thereby retaining the good quadrature properties of that integrator. Although we cannot give analytic justification, we believe our integrator is better because it has better quadrature. Prior developments of Rosenbrock integrators have neglected quadrature in their attempts to obtain optimum stability and/or truncation errors or to minimize function evaluations. It is not that these considerations are invalid; rather, it is that *engineering* an efficient and robust integrator requires making compromises between order of accuracy, stability, truncation errors, *and* quadrature.

TABLE 4.1: TOTAL NUMBER OF STEPS REQUIRED BY RUNGE-KUTTA AND ROSENBROCK-WOLFBRANDT METHODS

Example	Stiffness ratio, $\mathcal{O}\left(\frac{\lvert\lambda_{max}\rvert}{\lvert\lambda_{min}\rvert}\right)$	Tolerance, 10^{-p}	Integrators			
			EFA	SKA		FGA
				G.Err[a]	K.Err	
Brusselator (unstable)	$\mathcal{O}(10^1)$	5	1 487	12 206	22 460	175
		4	842	1 344	2 250	106
		3	483	211	234	66
	$\mathcal{O}(10^4)$	5	52 207	3 359	6 854	35
		4	26 533	416	707	26
		3	14 823	96	89	20
Lorenz chaos (unstable)	$\mathcal{O}(10^1)$	5	(b)	60 908	138 919	346
		4	(b)	6 418	13 905	208
		3	(b)	864	1 401	130
Enright's D-4 problem (singular)	∞	5	(b)	358	648	22
		4	(b)	80	86	19
		3	(b)	42[c]	34[c]	17
Viscoplastic model (stable)	$\mathcal{O}(10^5)$	5	89 186	17 848	54 970	287
		4	79 726	2 108	5 573	185
		3	39 906	427	635	134
Chemical kinetics (stable)	$\mathcal{O}(10^{13})$	5	(b)	605	1 131	115[c]
		4	(b)	170	162	116[c]
		3	(b)	88	60	95[c]

[a] Warning: More than 40 step reductions were needed to achieve the required tolerance in most solutions.

[b] Unrestrained error growth.

[c] Inaccurate results.

Appendix A

Derivation of Coefficients

A.1 Explicit Integrator

The first objective in the overall scheme of developing our Rosenbrock integrator was to add a fifth stage to Kutta's (1901), classic, *fourth-order*, explicit integrator. This addition was done so that a *third-order* embedded integrator of the Fehlberg (1969) type could be constructed for use in error estimation. No third-order integrator is embedded in a four-stage Runge-Kutta integrator; hence, the need for adding a fifth stage (Hairer et al., 1993, p. 167). A fifth-order error estimate would require two additional stages instead of just the one required by a third-order estimate.

The *order conditions* of Butcher (1963) for a fourth-order Runge-Kutta integrator in 4 stages comprise a set of 11 equations in 13 unknowns. Given the three constraint equations

$$\left.\begin{aligned} a_1 &= a_{10} \\ a_2 &= a_{20} + a_{21} \\ a_3 &= a_{30} + a_{31} + a_{32} \end{aligned}\right\}, \tag{A.1}$$

Butcher's set of eight order conditions is given by

$$\left.\begin{aligned} 1 &= c_0 + c_1 + c_2 + c_3 \\ \tfrac{1}{2} &= c_1 a_1 + c_2 a_2 + c_3 a_3 \\ \tfrac{1}{3} &= c_1 a_1^2 + c_2 a_2^2 + c_3 a_3^2 \\ \tfrac{1}{6} &= c_2 a_{21} a_1 + c_3 (a_{32} a_2 + a_{31} a_1) \\ \tfrac{1}{4} &= c_1 a_1^3 + c_2 a_2^3 + c_3 a_3^3 \\ \tfrac{1}{12} &= c_2 a_{21} a_1^2 + c_3 (a_{32} a_2^2 + a_{31} a_1^2) \\ \tfrac{1}{8} &= c_2 a_2 a_{21} a_1 + c_3 a_3 (a_{32} a_2 + a_{31} a_1) \\ \tfrac{1}{24} &= c_3 a_{32} a_{21} a_1 \end{aligned}\right\}. \tag{A.2}$$

The unknowns are the a_i, a_{ij}, and c_i, which are the parameters appearing in the Runge-Kutta formulæ of equations(2.9) and (2.11). The coefficients of Kutta (1901) displayed in table 2.1 represent one of many admissible solutions to this underconstrained system of equations. Kutta's formulæ are the foundation of our integrator. From here on, these coefficients are considered fixed.

To ensure that the integrator retains fourth-order accuracy after a fifth stage has been added, one must consider the extended set of order conditions

$$\left.\begin{aligned} 1 &= c_0 + c_1 + c_2 + c_3 + c_4 \\ \tfrac{1}{2} &= c_1 a_1 + c_2 a_2 + c_3 a_3 + c_4 a_4 \\ \tfrac{1}{3} &= c_1 a_1^2 + c_2 a_2^2 + c_3 a_3^2 + c_4 a_4^2 \\ \tfrac{1}{6} &= c_2 a_{21} a_1 + c_3 (a_{32} a_2 + a_{31} a_1) \\ &\quad + c_4 (a_{43} a_3 + a_{42} a_2 + a_{41} a_1) \\ \tfrac{1}{4} &= c_1 a_1^3 + c_2 a_2^3 + c_3 a_3^3 + c_4 a_4^3 \\ \tfrac{1}{12} &= c_2 a_{21} a_1^2 + c_3 (a_{32} a_2^2 + a_{31} a_1^2) \\ &\quad + c_4 (a_{43} a_3^2 + a_{42} a_2^2 + a_{41} a_1^2) \\ \tfrac{1}{8} &= c_2 a_2 a_{21} a_1 + c_3 a_3 (a_{32} a_2 + a_{31} a_1) \\ &\quad + c_4 a_4 (a_{43} a_3 + a_{42} a_2 + a_{41} a_1) \\ \tfrac{1}{24} &= c_3 a_{32} a_{21} a_1 \\ &\quad + c_4 \big(a_{43} (a_{32} a_2 + a_{31} a_1) + a_{42} a_{21} a_1\big) \end{aligned}\right\}. \tag{A.3}$$

The first two of these equations imply second-order accuracy, the second set of two equations implies third-order accuracy, and the last four equations imply fourth-order accuracy, as in equation(A.2). Because Kutta's original integrator is to be contained within this solution, one obtains the constraints

$$c_0 = \tfrac{1}{6}\,,\ c_1 = \tfrac{1}{3}\,,\ c_2 = \tfrac{1}{3}\,,\ c_3 = \tfrac{1}{6} \quad \Rightarrow \quad c_4 = 0 \tag{A.4}$$

and

$$\left.\begin{aligned} a_1 &= \tfrac{1}{2} \\ a_2 &= 0 + \tfrac{1}{2} \\ a_3 &= 0 + 0 + 1 \\ a_4 &= a_{40} + a_{41} + a_{42} + a_{43} \end{aligned}\right\}. \tag{A.5}$$

Furthermore, requiring $a_{4j} = c_j \ \forall\ j$, indicating an FSAL integrator, implies that $\mathbf{f}_{n+1} = \mathbf{k}_4$, which is needed by equation(2.3) for local interpolation.

FSAL integrators permit the assignment of $\mathbf{f}_n \Leftarrow \mathbf{f}_{n+1}$ in the next integration step, instead of calculating $\mathbf{f}_n$ directly, thereby eliminating one function evaluation per time step. As it turns out, any set of finite numbers for the a_{4j} satisfies the above equations because $c_4 = 0$, which is a coefficient to all terms containing the a_{4j}.

The order conditions for a third-order Runge-Kutta integrator in five stages, which is to be used in error analysis, are given by the following four equations in five unknowns (the a_i and a_{ij} are assumed known); in particular,

$$\left.\begin{aligned} 1 &= \widehat{c}_0 + \widehat{c}_1 + \widehat{c}_2 + \widehat{c}_3 + \widehat{c}_4 \\ \tfrac{1}{2} &= \widehat{c}_1 a_1 + \widehat{c}_2 a_2 + \widehat{c}_3 a_3 + \widehat{c}_4 a_4 \\ \tfrac{1}{3} &= \widehat{c}_1 a_1^2 + \widehat{c}_2 a_2^2 + \widehat{c}_3 a_3^2 + \widehat{c}_4 a_4^2 \\ \tfrac{1}{6} &= \widehat{c}_2 a_{21} a_1 + \widehat{c}_3 (a_{32} a_2 + a_{31} a_1) \\ &\quad + \widehat{c}_4 (a_{43} a_3 + a_{42} a_2 + a_{41} a_1) \end{aligned}\right\} \tag{A.6}$$

where the $\widehat{c}_i$ are the unknowns, of which $\widehat{c}_i \neq c_i$ for at least one i in order that the embedded integrator be different from the actual integrator. The quadrature of integration remains the same as Kutta's solution if one simply swaps $c_3 = 1/6$ with $c_4 = 0$ such that $\widehat{c}_3 = 0$ and $\widehat{c}_4 = 1/6$, with all other $\widehat{c}_i$ being equal to their c_i counterparts. This swapping is admissible (cf. Hairer et al., 1993, p. 167). Furthermore, for this to be third order, the fourth-order order conditions must be violated with this set of constants, which they are. These violated equations are precisely the truncation errors of the embedded integrator.

A.2 Semi-Implicit Integrator

Considering the constraint equations

$$\left.\begin{aligned} a_1 &= a_{10} \\ a_2 &= a_{20} + a_{21} \\ a_3 &= a_{30} + a_{31} + a_{32} \\ a_4 &= a_{40} + a_{41} + a_{42} + a_{43} \end{aligned}\right\} \tag{A.7}$$

and

$$\left.\begin{aligned} b_0 &= b \\ b_1 &= b_{10} + b \\ b_2 &= b_{20} + b_{21} + b \\ b_3 &= b_{30} + b_{31} + b_{32} + b \\ b_4 &= b_{40} + b_{41} + b_{42} + b_{43} + b \end{aligned}\right\} \tag{A.8}$$

along with the definitions

$$\left.\begin{aligned} g_{ij} &= a_{ij} + b_{ij} \\ g_i &= \sum_{j=0}^{i-1} (a_{ij} + b_{ij}) = a_i + b_i - b \end{aligned}\right\}, \tag{A.9}$$

the order conditions of Wolfbrandt (1978, pp. 96–100) for a fourth-order, modified, Rosenbrock integrator in five stages are therefore given by (cf. Nørsett and Wolfbrandt, 1979, and Hairer and Wanner, 1991, pp. 110–119)

$$\left.\begin{aligned} A &= c_0 + c_1 + c_2 + c_3 + c_4 \\ B &= c_1 g_1 + c_2 g_2 + c_3 g_3 + c_4 g_4 \\ C &= c_1 a_1^2 + c_2 a_2^2 + c_3 a_3^2 + c_4 a_4^2 \\ D &= c_2 g_{21} g_1 + c_3 (g_{32} g_2 + g_{31} g_1) \\ &\quad + c_4 (g_{43} g_3 + g_{42} g_2 + g_{41} g_1) \\ E &= c_1 a_1^3 + c_2 a_2^3 + c_3 a_3^3 + c_4 a_4^3 \\ F &= c_2 g_{21} a_1^2 + c_3 (g_{32} a_2^2 + g_{31} a_1^2) \\ &\quad + c_4 (g_{43} a_3^2 + g_{42} a_2^2 + g_{41} a_1^2) \\ G &= c_2 a_2 a_{21} g_1 + c_3 a_3 (a_{32} g_2 + a_{31} g_1) \\ &\quad + c_4 a_4 (a_{43} g_3 + a_{42} g_2 + a_{41} g_1) \\ H &= c_3 g_{32} g_{21} g_1 \\ &\quad + c_4 \big(g_{43} (g_{32} g_2 + g_{31} g_1) + g_{42} g_{21} g_1\big) \end{aligned}\right\} \tag{A.10}$$

wherein

$$\left.\begin{aligned} A &= 1 \\ B &= \tfrac{1}{2}(1 - 2b) \\ C &= \tfrac{1}{3} \\ D &= \tfrac{1}{6}(1 - 6b + 6b^2) \\ E &= \tfrac{1}{4} \\ F &= \tfrac{1}{12}(1 - 4b) \\ G &= \tfrac{1}{24}(3 - 8b) \\ H &= \tfrac{1}{24}(1 - 12b + 36b^2 - 24b^3) \end{aligned}\right\}. \tag{A.11}$$

Any set of a_i, a_{ij}, b, b_i, b_{ij}, and c_i that satisfies these equations is an admissible Rosenbrock integrator.

An assignment of $b = 1/4$ places b outside the region $[0.39434, 1.28057]$ for an A-stable, fourth-order, Rosenbrock integrator of the Wolfbrandt type in four stages (Kaps and Rentrop, 1979), which applies to our solution $\mathbf{x}$. Nevertheless, it is contained in the middle region of $[0.10567, 0.10727] \cup [0.20385, 0.25] \cup [0.39434, \infty)$, which are the domains of stability at infinity. Because b is located on one of these boundaries, it is stable but not dampened at infinity. An advantage in choosing a small value for b is that, in general, the smaller its value, the smaller the error constant will be (Kaps and Wanner, 1981).

Because the Rosenbrock integrator being derived is to explicitly contain the Runge-Kutta integrator of the prior section, it follows that the a_i, a_{ij}, and c_i are known *a priori*, and only the b_i and b_{ij} remain as unknowns, resulting in a system of 10 equations in 15 unknowns. The constraint equations therefore

become

$$\left.\begin{aligned} b_0 &= \tfrac{1}{4} \\ b_1 &= b_{10} + \tfrac{1}{4} \\ b_2 &= b_{20} + b_{21} + \tfrac{1}{4} \\ b_3 &= b_{30} + b_{31} + b_{32} + \tfrac{1}{4} \\ b_4 &= b_{40} + b_{41} + b_{42} + b_{43} + \tfrac{1}{4} \end{aligned}\right\}, \tag{A.12}$$

and the order conditions reduce to the set

$$\left.\begin{aligned} -1 &= 8b_1 + 8b_2 + 4b_3 \\ -\tfrac{1}{2} &= (1 + 2b_{21} + b_{31})(1 + 4b_1) \\ &\quad + (1 + b_{32})(1 + 4b_2) \\ -2 &= 2b_{21} + b_{31} + b_{32} \\ -1 &= 4b_1 + 8b_2 \\ -\tfrac{1}{4} &= (1 + b_{32})(1 + 2b_{21})(1 + 4b_1) \end{aligned}\right\}. \tag{A.13}$$

The first of these five order conditions is equivalent to the requirement that $\sum_{i=0}^{4} c_i b_i = 0$, so that "integrated" contributions from the nonautonomous gradients vanish (i.e., $h \sum_{i=0}^{4} c_i b_i \, \partial \mathbf{f}_n / \partial t = 0$) and cannot therefore affect adversely the stability of the solution.

As it turns out, one can express the coefficients b_{10}, b_{20}, b_{21}, b_{30}, b_{31}, and b_{32} all in terms of b_1, which is useful; specifically,

$$\left.\begin{aligned} b_{10} &= (-1 + 4b_1) \;/\; 4 \\ b_{20} &= (1 + 6b_1 - 8b_1^2) \;/\; 4(1 + 4b_1) \\ b_{21} &= -(5 + 28b_1) \;/\; 8(1 + 4b_1) \\ b_{30} &= (1 - 2b_1 - 8b_1^2) \;/\; 2(1 + 4b_1) \\ b_{31} &= (-3 + 8b_1 + 144b_1^2) \\ &\quad \div 4(1 + 4b_1)(1 + 12b_1) \\ b_{32} &= -12b_1 \;/\; (1 + 12b_1) \end{aligned}\right\}, \tag{A.14}$$

where it is apparent that $b_1 \neq -1/12$ and $b_1 \neq -1/4$. The simplest set of rational numbers that one can obtain is obviously acquired by taking $b_1 = 0$, which is what we did.

All that is left is one equation in five unknowns; it is

$$b_4 = b_{40} + b_{41} + b_{42} + b_{43} + \tfrac{1}{4} \,. \tag{A.15}$$

Certainly, a unique solution does not exist.

Constraints are sought from the four order conditions for a third-order Rosenbrock integrator in five stages to be used for error analysis; they are

$$\left.\begin{aligned} A &= \widehat{c}_0 + \widehat{c}_1 + \widehat{c}_2 + \widehat{c}_3 + \widehat{c}_4 \\ B &= \widehat{c}_1 g_1 + \widehat{c}_2 g_2 + \widehat{c}_3 g_3 + \widehat{c}_4 g_4 \\ C &= \widehat{c}_1 a_1^2 + \widehat{c}_2 a_2^2 + \widehat{c}_3 a_3^2 + \widehat{c}_4 a_4^2 \\ D &= \widehat{c}_2 g_{21} g_1 + \widehat{c}_3 (g_{32} g_2 + g_{31} g_1) \\ &\quad + \widehat{c}_4 (g_{43} g_3 + g_{42} g_2 + g_{41} g_1) \end{aligned}\right\}. \tag{A.16}$$

With all variables fixed except those in equation(A.15), the first and third equations in equation(A.16) are satisfied identically. From the second equation comes the requirement that $b_4 = 0$. Therefore, one obtains the following two equations in four unknowns:

$$\left.\begin{aligned} -\tfrac{1}{4} &= b_{40} + b_{41} + b_{42} + b_{43} \\ -5 &= 4b_{41} + 2b_{42} + 12b_{43} \end{aligned}\right\}. \tag{A.17}$$

Seeking simplicity, we chose the set of coefficients presented in table 2.2. They have the intuitive property that the two contributions at the quadrature points $a = 1/2$ and $a = 1$ cancel one another out.

Appendix B

Robinson's Viscoplastic Model

The system of differential equations that govern viscoplasticity, as derived by Robinson and Swindeman (1982), is given by

$$\left.\begin{aligned} \dot{\sigma} &= \mathbf{D}^{el}\cdot\left(\dot{\epsilon}-\dot{\epsilon}^I-\mathbf{I}\alpha\dot{T}\right)\\ \dot{\epsilon}^I &= \frac{p(x)\langle F\rangle^n}{2\bar{r}K_T\sqrt{J_2}}\,\boldsymbol{\Sigma}\\ \dot{\mathbf{A}} &= \frac{H}{G^\beta}\,\dot{\epsilon}^I_{ij}-\frac{\bar{R}G^{(m-\beta)}}{K_T\sqrt{I_2}}\,\mathbf{A}\end{aligned}\right\}, \qquad \text{(B.1)}$$

where $\boldsymbol{\sigma}$ denotes Cauchy stress, $\boldsymbol{\epsilon}^I$ denotes inelastic strain, and $\mathbf{A}$ denotes an internal or back stress. Quantities used to describe the parameters in these equations include

$$\mathbf{S}=\boldsymbol{\sigma}-(\tfrac{1}{3}\mathbf{I}:\boldsymbol{\sigma})\mathbf{I} \qquad \boldsymbol{\Sigma}=\mathbf{S}-\mathbf{A}$$
$$J_2=\tfrac{1}{2}(\boldsymbol{\Sigma}:\boldsymbol{\Sigma}) \qquad I_2=\tfrac{1}{2}(\mathbf{A}:\mathbf{A})$$

where $\mathbf{S}$ is the stress deviator.

The step function introduced by Robinson (1978) was later represented by Robinson and Swindeman (1982) with the spline function

$$p(x)=\begin{cases}0 & \text{if } x<-1\\ (1+x)^2/2 & \text{if } -1\le x\le 0\\ 1-(1-x)^2/2 & \text{if } 0\le x\le 1\\ 1 & \text{if } x>1\end{cases}$$

whose argument is given by

$$x=\begin{cases}(\mathbf{S}:\mathbf{A})/w_2 & \text{in } G\\ (\mathbf{S}:\boldsymbol{\Sigma})/w_1 & \text{in } \dot{\epsilon}^I\,.\end{cases}$$

The scalar-valued state functions are given by

$$G'=\begin{cases}G_b & \text{for } G_b\ge 2G^0\\ G_b^2/4G^0+G^0 & \text{for } G<2G^0\end{cases}$$

with

$$G_b(I_2)=\frac{\sqrt{I_2}}{K_T}$$

and by

$$G(I_2)=(G'-G^0)p(x)+G^0$$

$$F(J_2)=\begin{cases}0 & \text{if } \sqrt{J_2}/K_T-1\le 0\\ \sqrt{J_2}/K-1 & \text{if } \sqrt{J_2}/K_T-1>0\,.\end{cases}$$

B.1 Material Constants

The material constants for a 2-$\frac{1}{4}$ wt% chromium–1 wt% molybdenum steel given in Robinson and Swindeman (1982) are

Reference temperature	T_0	=	1000
Young's modulus	E	=	22 480
Poisson's ratio	ν	=	0.334
Thermal coefficient	α	=	0.1×10^{-5}
Drag strength	K_T	=	0.82
Yield exponent	n	=	4
Flow factor	r	=	3.61×10^{7}
Recovery power	m	=	7.73
Hardening power	β	=	1.5
Recovery coefficient	R	=	9×10^{-8}
Hardening coefficient	H	=	1.37×10^{-4}
Back stress threshold	G_0	=	0.1
Step function range	w_1	=	1
Step function range	w_2	=	1

The temperature-dependent parameters are

$$\bar{r}=r\exp\left[(23.8T-2635)(1/T_0-1/T)\right]$$
$$\bar{R}=R\exp\left[40\,000(1/T_0-1/T)\right]$$

The elastic moduli are isotropic, that is,

$$\nu=\frac{E}{2(\nu+1)} \qquad \lambda=\frac{E\nu}{(1-2\nu)(1+\nu)}$$

with stiffness matrix $\mathbf{D}^{el}$ of

$$\begin{bmatrix} \lambda+2\nu & -2\nu & -2\nu & 0 & 0 & 0 \\ -2\nu & \lambda+2\nu & -2\nu & 0 & 0 & 0 \\ -2\nu & 0 & \lambda+2\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\nu \end{bmatrix}$$

when expressed in Voigt notation.

B.2 Code Implementation

The governing system of differential equations given in equation(B.1) can be consolidated into two, tensor, differential equations and a scalar function evaluation; in particular,

$$\dot{\mathbf{A}} = \frac{2rH}{G^{\beta}}\,\dot{C}_{\Sigma}\,\boldsymbol{\Sigma} - \dot{C}_A\,\mathbf{A}$$

$$\dot{\boldsymbol{\Sigma}} = -\dot{C}_{\Sigma}\,\mathbf{D}^{el}\cdot\boldsymbol{\Sigma} - \dot{\mathbf{A}} + \mathbf{D}^{el}\cdot\left[\dot{\boldsymbol{\epsilon}} - \frac{1}{3}(\mathbf{I}:\dot{\boldsymbol{\epsilon}})\mathbf{I}\right]$$

and

$$H_{ydro} = (3\lambda+2\nu)\frac{(\mathbf{I}:\dot{\boldsymbol{\epsilon}}) - 3\alpha\dot{T}}{3}\,,$$

wherein

$$\dot{C}_{\Sigma} = \frac{p(x)\langle F\rangle^n}{2\bar{r}\sqrt{J_2}} \qquad \dot{C}_A = \frac{\bar{R}\,G^{m-\beta}}{\sqrt{I_2}}\,.$$

B.2.1 The Jacobian

The Jacobian is composed of a 2×2 matrix with each element being a fourth-order tensor.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial\dot{\boldsymbol{\Sigma}}}{\partial\boldsymbol{\Sigma}} & \frac{\partial\dot{\boldsymbol{\Sigma}}}{\partial\mathbf{A}} \\ \frac{\partial\dot{\mathbf{A}}}{\partial\boldsymbol{\Sigma}} & \frac{\partial\dot{\mathbf{A}}}{\partial\mathbf{A}} \end{bmatrix}$$

Its components are defined according to

$$\frac{\partial\dot{\boldsymbol{\Sigma}}}{\partial\boldsymbol{\Sigma}} = -2\mu\,\boldsymbol{\Sigma}\cdot\frac{\partial\dot{C}_{\Sigma}}{\partial\boldsymbol{\Sigma}} - \frac{\partial\dot{\mathbf{A}}}{\partial\boldsymbol{\Sigma}} - 2\mu\,\dot{C}_{\Sigma}\,\mathbf{I}$$

$$\frac{\partial\dot{\boldsymbol{\Sigma}}}{\partial\mathbf{A}} = -2\mu\,\boldsymbol{\Sigma}\cdot\frac{\partial\dot{C}_{\Sigma}}{\partial\mathbf{A}} - \frac{\partial\dot{\mathbf{A}}}{\partial\mathbf{A}}$$

$$\frac{\partial\dot{\mathbf{A}}}{\partial\boldsymbol{\Sigma}} = \frac{2rH}{G^{\beta}}\left[\boldsymbol{\Sigma}\cdot\left(\frac{-\beta}{G}\dot{C}_{\Sigma}\,\frac{\partial G}{\partial\boldsymbol{\Sigma}} + \frac{\partial\dot{C}_{\Sigma}}{\partial\boldsymbol{\Sigma}}\right) + \dot{C}_{\Sigma}\,\mathbf{I}\right] - \mathbf{A}\cdot\frac{\partial\dot{C}_A}{\partial\boldsymbol{\Sigma}}$$

$$\frac{\partial\dot{\mathbf{A}}}{\partial\mathbf{A}} = \frac{2rH}{G^{\beta}}\,\boldsymbol{\Sigma}\cdot\left(\frac{-\beta}{G}\dot{C}_{\Sigma}\,\frac{\partial G}{\partial\mathbf{A}} + \frac{\partial\dot{C}_{\Sigma}}{\partial\mathbf{A}}\right) - \mathbf{A}\cdot\frac{\partial\dot{C}_A}{\partial\mathbf{A}} - \dot{C}_A\,\mathbf{I}\,,$$

which themselves contain the derivatives

$$\frac{\partial\dot{C}_{\Sigma}}{\partial\boldsymbol{\Sigma}} = \frac{\langle F^n\rangle}{2\bar{r}\sqrt{J_2}}\left[\frac{\partial p(x)}{\partial x}\,\frac{\partial x}{\partial\boldsymbol{\Sigma}} + p(x)\left(\frac{n}{I_2K_T^2} - \frac{1}{2J_2}\right)\right]$$

$$\frac{\partial\dot{C}_{\Sigma}}{\partial\mathbf{A}} = \frac{\langle F^n\rangle}{2\bar{r}\sqrt{J_2}}\,\frac{\partial p(x)}{\partial x}\,\frac{\partial x}{\partial\mathbf{A}}$$

$$\frac{\partial\dot{C}_A}{\partial\boldsymbol{\Sigma}} = \frac{\dot{C}_A(m-\beta)}{G}\,\frac{\partial G}{\partial\boldsymbol{\Sigma}}$$

$$\frac{\partial\dot{C}_A}{\partial\mathbf{A}} = \frac{\dot{C}_A(m-\beta)}{G}\frac{\partial G}{\partial\mathbf{A}} - \frac{\dot{C}_A}{2\sqrt{I_2}}\,\mathbf{A}\,,$$

wherein

$$\frac{\partial p(x)}{\partial x} = \begin{cases} 0 & \text{if } x < -1 \\ 1+x & \text{if } -1 \le x \le 0 \\ 1-x & \text{if } 0 \le x \le 1 \\ 0 & \text{if } x > 1 \end{cases}$$

and

$$\frac{\partial x}{\partial\boldsymbol{\Sigma}} = \begin{cases} (\mathbf{A}+2\boldsymbol{\Sigma})/w_1 \\ \mathbf{A}/w_2 \end{cases}$$

$$\frac{\partial x}{\partial\mathbf{A}} = \begin{cases} \boldsymbol{\Sigma}/w_1 \\ (\boldsymbol{\Sigma}+2\mathbf{A})/w_2 \end{cases}\,.$$

References

Arnold, S. M.: Quantification of Numerical Stiffness for a Unified Viscoplastic Constitutive Model. *J. Eng. Mater. Tech.*, vol. 112, 1990, pp. 271–276.

Butcher, J. C.: Coefficients for the Study of Runge-Kutta Integration Processes. *J. Aust. Math. Soc.*, vol. 3, 1963, pp. 185–201.

Enright, W. H.; and Pryce, J. D.: Two FORTRAN Packages for Assessing Initial-Value Methods. *ACM Trans. Math. Softw.*, vol. 13, 1987, pp. 1–27.

Fehlberg, E.: Low-Order Classical Runge-Kutta Formulas With Stepsize Control and Their Application to Some Heat Transfer Problems. NASA TR R–315, 1969.

Gustafsson, K.; Lundh, M.; and Söderlind; G.: A PI Stepsize Control for the Numerical Solution of Ordinary Differential Equations. *BIT*, vol. 28, no. 2, 1988, pp. 270–287.

Hairer, E.; Nørsett, S. P.; and Wanner, G.: *Solving Ordinary Differential Equations I: Nonstiff Problems.* Second ed., Springer Series in Computational Mathematics, vol. 8, R. L. Graham, J. Stoer, and R. Varga, eds., Springer-Verlag, Berlin, 1993.

Hairer, E; and Wanner, G.: *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems.* Springer Series in Computational Mathematics, vol. 14, G. Wanner, ed., Springer-Verlag, Berlin, 1991.

Kaps, P.; Poon, S. W. H; and Bui, T. D.: Rosenbrock Methods for Stiff ODEs: A Comparison of Richardson Extrapolation and Embedding Technique. *Co.*, vol. 34, 1985, pp. 17–40.

Kaps, P.; and Rentrop, P.: Generalized Runge-Kutta Methods of Order 4 With Stepsize Control for Stiff Ordinary Differential-Equations. *Numer. Math.*, vol. 33, 1979, pp. 55–68.

Kaps, P.; and Wanner, G.: A Study of Rosenbrock-Type Methods of High-Order. *Numer. Math.*, vol. 38, 1981, pp. 279–298.

Kutta, W.: Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. *Z. Math. Phys.*, vol. 46, 1901, pp. 435–453.

Lambert, J. D.: *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem.* John Wiley & Sons, Chichester, England, 1991.

Lorenz, E. N.: Deterministic Nonperiodic Flow. *J. Atmos. Sci.*, vol. 20, 1963, pp. 130–141.

Nørsett, S. P.; and Wolfbrandt, A.: Order Conditions for Rosenbrock Type Methods. *Numer. Math.*, vol. 32, 1979, pp. 1–15.

Press, W. H., et al.: *Numerical Recipes in FORTRAN: The Art of Scientific Computing.* Second ed., Cambridge University Press, Cambridge, England, 1992.

Robinson, D. N.: Unified Creep-Plasticity Model for Structural Metals at High Temperature. ORNL TM–5969, 1978.

Robinson, D. N.; and Swindeman, R.: Unified Creep-Plasticity Constitutive Equations for 2-1/4 CR–1 Mo Steel at Elevated Temperature. ORNL TM–8444, 1982.

Rosenbrock, H. H.: Some General Implicit Processes for the Numerical Solution of Differential Equations. *Comput. J.*, vol. 5, 1963, pp. 329–330.

Shampine, L. F.: Lipschitz Constants and Robust ODE Codes. *Computational Methods in Nonlinear Mechanics*, Proceedings of the TICOM Second International Conference, J. T. Oden, ed., North-Holland Publishing Company, New York, 1980, pp. 427–449.

Shampine, L. F.: Implementation of Rosenbrock Methods. *ACM Trans. Math. Softw.*, vol. 8, 1982, pp. 93–113.

Shampine, L. F.: Interpolation for Runge-Kutta Methods. *SIAM J. Num.*, vol. 22, 1985, pp. 1014–1027.

Shampine, L. F.; and Watts, H. A.: The Art of Writing a Runge-Kutta Code. II. *Appl. Math. Comput.*, vol. 5, 1979, pp. 93–121.

Wolfbrandt, A.: *A Study of Rosenbrock Processes With Respect to Order Conditions and Stiff Stability.* Ph.D. thesis, Chalmers University of Technology, Göteborg, Sweden, 1977.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1996	Technical Memorandum

4. TITLE AND SUBTITLE

Development and Applications of a Rosenbrock Integrator

5. FUNDING NUMBERS

WU-505-63-5A

6. AUTHOR(S)

Alan D. Freed and Ilana S. Iskovitz

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E-9809

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-4709

11. SUPPLEMENTARY NOTES

Alan D. Freed, NASA Lewis Research Center, and Ilana S. Iskovitz, National Research Council—NASA Research Associate at Lewis Research Center. Responsible person, Alan D. Freed, organization code 5110, (216) 433-8747.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 64

This publication is available from the NASA Center for Aerospace Information, (301) 621-0390.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

Rosenbrock (1963) extended Runge-Kutta integrators by introducing the Jacobian into the integrator, thereby making it implicit. Later, Wolfbrandt (1977) modified Rosenbrock's approach to produce more computationally efficient implementations. This report presents a Rosenbrock-Wolfbrandt integrator that is a direct extension of "the" Runge-Kutta integrator developed by Kutta (1901). A fifth stage has been added to produce an embedded third-order integrator for error estimation. The proposed integrator is applied to five different problems that are, to varying degrees, either stiff, unstable, or singular. These problems originated as mathematical models of physical phenomena. The integrator developed herein is compared against state-of-the-art Runge-Kutta and Rosenbrock integrators for efficiency at given accuracies and is shown to be an excellent numerical integrator.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Numerical integration; Differential equations	42
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

National Aeronautics and
Space Administration

Lewis Research Center
21000 Brookpark Rd.
Cleveland, OH 44135-3191